17673

NATIONAL LIBRARY
OTTAWA

CANADA

BIBLIOTHÈQUE NATIONALE
OTTAWA

NAME OF AUTHOR.. HARTMUT W. ROSEKE

TITLE OF THESIS.... NORDSIECK'S METHOD

UNIVERSITY..... OF ALBERTA

DEGREE FOR WHICH THESIS WAS PRESENTED.... M. Sc.

YEAR THIS DEGREE GRANTED..... 1973

Permission is hereby granted to THE NATIONAL LIBRARY OF CANADA to microfilm this thesis and to lend or sell copies of the film.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

(Signed) H. W. Roseke

PERMANENT ADDRESS:

11815-91 ST.
EDMONTON,
ALTA.

DATED MAY 31, 1973

NL-91 (10-68)

THE UNIVERSITY OF ALBERTA

NORDSIECK'S METHOD

by

© HARTMUT W. ROSEKE

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTING SCIENCE

EDMONTON, ALBERTA

FALL 1973

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled NORDSIECK'S METHOD, submitted by Hartmut W. Roseke, in partial fulfillment of the requirements for the degree of Master of Science.

L. W. Jackson
Supervisor

J. Tolan

J. H. Edwards

John Tostan

Date May 18, 73

ABSTRACT

This thesis is concerned with the numerical approximation of initial value problems in differential equations.

A detailed development of Nordsieck's method is presented. The errors introduced by stepsize changes are then examined and a procedure to reduce these errors is introduced. Theoretical and practical considerations showed that the procedure is effective in reducing errors.

ACKNOWLEDGEMENTS

I wish to express my gratitude to Dr. L.W. Jackson for assisting in and supervising the preparation of this thesis. Dr. S. Cabay has also read this thesis and has provided helpful suggestions.

TABLE OF CONTENTS

		Page
CHAPTER I	INTRODUCTION	1
CHAPTER II	NORDSIECK'S METHOD	4
2.1	Notation	4
2.2	The Predictor	5
2.3	The Corrector	10
2.4	Parameter Definitions	19
2.5	Truncation Error	26
CHAPTER III	STEPSIZE AND ORDER	29
3.1	Introduction	29
3.2	Stepsize-Changing Errors	30
3.3	Tests for Stepsize and Order	37
3.4	A Stepsize Changing Algorithm	42
CHAPTER IV	OUTLINE OF THE COMPUTER PROGRAM	47
4.1	Introduction	47
4.2	General Description of the Program	48
4.3	The Arguments of NDSK	51
4.4	The Driving Function FEVAL	53
4.5	The Tests	54
4.6	The History of the Program	56

		Page
CHAPTER V	RESULTS OF THE TEST PROBLEMS	58
5.1	Introduction and Summary	58
5.2	Equations of Exponential Functions	60
5.3	Sine-Cosine Differential Equations	62
5.4	A Non-Linear Problem	63
5.5	The Satellite Orbit Problem	65
5.6	A Stiff Linear System	67
5.7	The Solution of $y^{(1)} = 20y/t$	68
5.8	A Problem with Discontinuous f	69
5.9	The Solution of $y^{(1)} = -2ty^2$	71
5.10	Solve for y in $y(t) = \int_0^{\pi} \sin(t \sin z)\, dz$	72
5.11	Problems with Singularities	73
5.12	The Hull and Cremer Problems	75
5.13	The Solution of $y^{(1)} = y \log y/t$	78
5.14	The Solution of a Bessel Equation	79
5.15	The Solution of $y^{(1)} = a^2b/(t^2+a^2)$	81
5.16	The Solutions of ODE's with Error Estimates	83
5.17	The Parabolic Surface	85
5.18	The Ricatti-Bessel Equation	87
5.19	An Oscillating Solution	91
5.20	An Unstable Problem	93
CHAPTER VI	CONCLUDING REMARKS	95
REFERENCES AND BIBLIOGRAPHY		97
APPENDIX	THE PROGRAM LISTINGS	100

CHAPTER I

INTRODUCTION

The objective of this thesis is to study Nordsieck's method for solving the differential equation

$$y^{(1)} = f(t,y) , \tag{1.1.1}$$

given $y(t_0) = y_0$.

Nordsieck's method consists of a predictor equation of the form

$$\underline{x}_{n+1}^{(0)} = Q\underline{x}_n , \tag{1.1.2}$$

and a corrector equation of the form

$$\underline{x}_{n+1}^{(m+1)} = \underline{x}_{n+1}^{(m)} + F_{n+1}^{(m)} \underline{\ell} , \quad m=0,1,2,\ldots,M-1 , \tag{1.1.3}$$

where

$$\underline{x}_n = \mathrm{col}(y_n, hy_n^{(1)}, hy_n^{(2)}/2!, \ldots, h^q y_n^{(q)}/q!) . \tag{1.1.4}$$

Here, $y_n^{(j)}$ is the j-th derivative of a polynomial that approximates the solution about a point t_n. The matrix Q and vector $\underline{\ell}$ are chosen so that the truncation error is low and the method is stable. The quantities $F_{n+1}^{(m)}$

are scalars that measure how well the differential equation is satisfied.

One of the principal advantages of Nordsieck's method is that estimates of local errors are easily obtainable. In fact, when a q-th order method has been used to make a step then the error for the (q-1)-th order method is immediately available since all derivatives from the first to the q-th are available; and by taking differences estimates of the errors for the q-th and (q+1)-th order methods can easily be computed.

Because local estimates are readily available it is possible to choose both order and stepsize to minimize, at least locally, the cost of solving differential equations for a given tolerance.

An implementation of Nordsieck's method with facilities for changing the stepsize and order has been written and is included in the Appendix. While experimenting with this program we observed that large errors are introduced into the numerical solution when using methods of order greater than 5. Similar observations are also reported by Krough [15]. Hence, we developed and included in the program a procedure to substantially lower the stepsize changing errors when the stepsize decreases.

In Chapter II the predictor and corrector equations of Nordsieck's method are developed using matrix notation. Matrix notation simplifies the analysis considerably, it illustrates the method's relation to multistep methods more readily, and it leads quite easily to a more general class of integration methods called multivalue methods by Gear [7].

In Chapter III stepsize changing is discussed. In particular, different methods of obtaining local truncation error estimates are considered. Also, we present the development of a technique to correct the solution when decreasing the stepsize.

In Chapter IV the implementation of Nordsieck's method is discussed. Chapter V contains the test problems.

CHAPTER II

NORDSIECK'S METHOD

2.1 Notation

Let y be the true solution and Y be the approximate solution of (1.1.1). We use the abbreviations

$$y_n = y(t_n) \quad ,$$

$$Y_n = Y(t_n) \quad ,$$

and $$y_n^{(1)} = f(t_n, Y_n) \quad ,$$

$$Y_n^{(1)} = f(t_n, Y_n) \quad ,$$

$$= f_n \quad ,$$

where $t_n = t_{n-1} + h$, $n=1,2,\ldots,N$, t_0 given, and h is the stepsize.

Vectors are denoted by underlined lower case letters. The symbol $||.||$ is used for the Euclidean vector norm. For example,

$$\underline{x}^{(1)}(t) = A\,\underline{x}(t)$$

is a constant coefficient linear system, $x(t)_i$ is the i-th component of the solution vector $\underline{x}(t)$, and $||\underline{x}(t)||^2 = \sum (x(t)_i)^2$ is the square of the Euclidean norm. Moreover, if $\underline{x}_1, \underline{x}_2, \underline{x}_3, \ldots, \underline{x}_n$ is a sequence

of vectors, then $x_{n,i}$ denotes the i-th component of the n-th vector in the sequence.

For convenience zero indexing is used for vectors and matrices.

2.2 The Predictor

To develop the predictor equation (1.1.2) of Nordsieck's method we consider the truncated Taylor series

$$Y_{n+1} = Y_n + h\, Y_n^{(1)} + \frac{h^2}{2!} Y_n^{(2)} + \ldots + \frac{h^q}{q!} Y_n^{(q)} , \tag{2.2.1}$$

assuming that the approximate solution Y has derivatives of sufficiently high orders over the interval $[t_n, t_{n+1}]$.

The predictor can be formulated in matrix notation by using the vectors

$$\underline{z}_n = \mathrm{col}(Y_n, Y_n^{(1)}, \ldots, Y_n^{(q)}) \tag{2.2.2}$$

and

$$\underline{x}_n = \mathrm{col}(Y_n, hY_n^{(1)}, \ldots, \frac{h^q}{q!} Y_n^{(q)}) . \tag{2.2.3}$$

Relationships between $\underline{x}_n$ and $\underline{x}_{n+1}$, $\underline{z}_n$ and $\underline{z}_{n+1}$, and $\underline{x}_n$ and $\underline{z}_n$, can be obtained by expanding the components of $\underline{x}_{n+1}$ and $\underline{z}_{n+1}$ in truncated Taylor series about the point $t = t_n$. That is, by using

$$y_{n+1}^{(i)} = \sum_{j=i}^{q} \frac{h^{j-i}}{(j-i)!} y_n^{(j)} \quad , \quad i=0,1,2,\ldots,q \ , \qquad (2.2.4)$$

which may be rewritten as

$$\frac{h^i}{i!} y_{n+1}^{(i)} = \sum_{j=i}^{q} \binom{j}{i} \frac{h^j}{j!} y_n^{(j)} \ . \qquad (2.2.5)$$

Writing (2.2.4) and (2.2.5) in matrix notation yields

$$\underline{z}_{n+1} = \Phi(h)\underline{z}_n \ , \qquad (2.2.6)$$

and

$$\underline{x}_{n+1} = Q\,\underline{x}_n \ , \qquad (2.2.7)$$

where

$$\Phi(h)_{ij} = \begin{cases} \dfrac{h^{j-i}}{(j-i)!} \ , & \text{for } j \geq i \ , \\ 0 \ , & \text{for } j < i \ , \end{cases} \qquad (2.2.8)$$

and

$$Q_{ij} = \begin{cases} \binom{j}{i} \ , & \text{for } j \geq i \ , \\ 0 \ , & \text{for } j < i \ . \end{cases} \qquad (2.2.9)$$

Moreover, let

$$D(h)_{ij} = \frac{h^i}{i!} \delta_{ij} \ , \qquad (2.2.10)$$

δ_{ij} being the Kronecker delta; then, one can see that

$$Q = D(h)\Phi(h)D^{-1}(h) \qquad (2.2.11)$$

holds. Hence it is clear that if $\underline{x}_0 = D(h)\underline{z}_0$, then the sequences defined by (2.2.6) and (2.2.7) satisfy $\underline{x}_n = D(h)\underline{z}_n$. Thus, the use of (2.2.7) is equivalent to the use of (2.2.6). Furthermore, if $\underline{z}_n$ consists of the derivatives of a q-th degree polynomial evaluated at $t = t_n$, then $\underline{z}_{n+1}$ consists of the derivatives of the same polynomial evaluated at $t = t_{n+1}$. Recalling that the derivatives of a polynomial evaluated at a point uniquely define the polynomial, we see that $\underline{z}_n$ and $\underline{z}_{n+1}$ are simply different representations of the same polynomial defined by

$$P(t) = \sum_{j=0}^{q} \frac{(t - t_n)^j}{j!} z_{n,j} \quad . \qquad (2.2.12)$$

Since the sequence $\underline{z}_0, \underline{z}_1, \ldots$ simply gives different representations of the polynomial defined by $\underline{z}_0$, it should be clear that the use of the predictor alone cannot yield good approximations to the solution of a differential equation. Example 2.1.1 illustrates the folly of using the predictor alone.

Example 2.1.1

Use (2.2.7) to integrate

$$y^{(1)} = -\sin(t), \quad y(0) = 1 , \qquad (2.2.13)$$

with h = 0.1, and q = 4, over the interval [0,5].

Results of $y^{(1)} = -\sin(t)$, $y(0) = 1$

t	Y_n	y_n	Error
0.0	1.0	1.0	0.0
0.5	0.87760	0.87758	0.00002
1.0	0.54167	0.54030	0.00136
2.0	-0.33333	-0.41615	0.08281
3.0	-0.125	-0.98999	0.86499
4.0	3.66667	-0.65364	4.32031
5.0	14.54617	0.28366	14.25800

Table 2.2.1

The results of Table 2.2.1 were computed with the starting vector $\underline{x}_0$ obtained from the true solution y(t)= cos(t). That is

$$\underline{x}_0 = D(h)\underline{z}_0 ,$$

$$= D(h)\text{col}(1,0,-1,0,1) , \qquad (2.2.14)$$

which defines the polynomial

$$P(t) = 1 - \frac{t^2}{2} + \frac{t^4}{24} \quad . \tag{2.2.15}$$

Using this polynomial one can obtain the second column of Table 2.2.1. For example, P(5.0) = 14.54617.

An alternate method of coming to the same conclusion uses the fact that $\Phi(h)$ is a discrete transition matrix having the properties

$$\Phi(nh) = \Phi^n(h) \quad , \tag{2.2.16}$$

and

$$\Phi(-h) = \Phi^{-1}(h) \quad . \tag{2.2.17}$$

Thus, we can write

$$\begin{aligned} \underline{z}_n &= \Phi^n(h)\underline{z}_0 \quad , \\ &= \Phi(nh)\underline{z}_0 \quad , \end{aligned} \tag{2.2.18}$$

and

$$\begin{aligned} \underline{x}_n &= D(h)\underline{z}_n \quad , \\ &= D(h)\Phi(nh)\underline{z}_n \quad , \\ &= D(h)\Phi(nh)D^{-1}(h)\underline{x}_0 \quad , \end{aligned}$$

$$= D(h)\begin{pmatrix} 1 & nh & (nh)^2/2! & (nh)^3/3! & (nh)^3/4! \\ 0 & 1 & nh & (nh)^2/2! & (nh)^3/3! \\ 0 & 0 & 1 & nh & (nh)^2/2! \\ 0 & 0 & 0 & 1 & nh \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} D^{-1}(h)\underline{x}_o ,$$

$$= \begin{pmatrix} 1 & n & n^2 & n^3 & n^4 \\ 0 & 1 & 2n & 3n^2 & 4n^3 \\ 0 & 0 & 1 & 3n & 6n^2 \\ 0 & 0 & 0 & 1 & 4n \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \underline{x}_o \quad . \qquad (2.2.19)$$

Clearly, (2.2.19) shows that for almost all choices of $\underline{x}_o$ large errors in $\underline{x}_n$ may be expected.

2.3 The Corrector

The corrector of Nordsieck's method is of the form

$$\underline{x}_{n+1}^{(m+1)} = \underline{x}_{n+1}^{(m)} + F_{n+1}^{(m)}\,\underline{\ell} \;, \quad m=0,1,\ldots,M-1 \quad . \qquad (2.3.1)$$

Here, $F_{n+1}^{(m)}$ is a residual defined by

$$F_{n+1}^{(m)} = \begin{cases} hf_{n+1}^{(0)} - x_{n+1,1}^{(0)} & , \text{ for } m = 0 \ , \\ hf_{n+1}^{(m)} - x_{n+1,1}^{(m-1)} & , \text{ for } m > 0 \ , \end{cases} \tag{2.3.2}$$

where $f_{n+1}^{(m)} = f(t_{n+1}, Y_{n+1}^{(m)})$, and $\underline{\ell}$ is a vector of parameters which determines different methods. We set $\underline{x}_{n+1} = \underline{x}_{n+1}^{(M)}$, and $\underline{x}_{n+1}^{(0)} = Q\,\underline{x}_n$.

Equation (2.3.1) may also be written in the form

$$\underline{x}_{n+1}^{(m+1)} = Q\,\underline{x}_n + F_{n+1}^{*(m)}\,\underline{\ell} \ , \tag{2.3.3}$$

where

$$F_{n+1}^{*(m)} = hf_{n+1}^{(m)} - x_{n+1,0}^{(0)} \ . \tag{2.3.4}$$

Provided that (2.3.1) converges the corrector may be written as the implicit equation

$$\underline{x}_{n+1} = Q\,\underline{x}_n + F_{n+1}\,\underline{\ell} \ , \tag{2.3.5}$$

where

$$F_{n+1} = hf_{n+1} - x_{n+1,1}^{(0)} \ , \tag{2.3.6}$$

which is to be solved for $\underline{x}_{n+1}$.

The form of the corrector equation (2.3.1) may be motivated by the following analysis.

Let

$$x^T_{n,i} = h^i y^{(i)}_n / i! \ , \quad i=0,1,2,\ldots, \tag{2.3.7}$$

be the i-th scaled derivative of the true solution. Define the predictor error $\underline{\varepsilon}^{(0)}_{n+1}$ by

$$\underline{\varepsilon}^{(0)}_{n+1} = \underline{x}^T_{n+1} - \underline{x}^{(0)}_{n+1} \ , \tag{2.3.8}$$

and the error $\underline{\varepsilon}_{n+1}$ of the predictor-corrector method by

$$\underline{\varepsilon}_{n+1} = \underline{x}^T_{n+1} - \underline{x}_{n+1} \ . \tag{2.3.9}$$

Expanding the true solution y and its first q derivatives about the point t_n and formulating the resulting expressions in matrix notation as has been done to develop the predictor equation (2.2.7) we obtain

$$\underline{x}^T_{n+1} = Q \, \underline{x}^T_n + \underline{e} \ , \tag{2.3.10}$$

where

$$e_i = \sum_{j=q-i+1}^{\infty} \frac{h^{i+j}}{i!j!} y^{(i+j)}_n \ , \quad i=0,1,2,\ldots,q \ . \tag{2.3.11}$$

Subtracting the equation

$$\underline{x}^{(0)}_{n+1} = Q \, \underline{x}_n \tag{2.3.12}$$

from equation (2.3.10) and using the error definitions (2.3.8) and (2.3.9) yields

$$\underline{\varepsilon}_{n+1}^{(0)} = Q\,\underline{\varepsilon}_n + \underline{e} \quad . \tag{2.3.13}$$

If $\underline{\varepsilon}_n = 0$, then $\underline{\varepsilon}_{n+1}^{(0)} = \underline{e}$ and then, using (2.3.11),

$$\underline{\varepsilon}_{n+1}^{(0)} = \varepsilon_{n+1,0}^{(0)}\,\underline{c} + O(h^{q+2}) \quad , \tag{2.3.14}$$

where $c_i = \binom{q+1}{i}$, $i=0,1,2,\ldots,q$. Thus, for small h, $\varepsilon_{n+1,0}^{(0)}\,\underline{c}$ is a good approximation to $\underline{\varepsilon}_{n+1}^{(0)}$.

In Corollary 2.3.3 to follow it is shown that

$$\varepsilon_{n+1,0}^{(0)} = (q+1)^{-1}\,F_{n+1}^{(0)} + O(h^{q+2}) \quad . \tag{2.3.15}$$

Moreover, in Corollary 2.3.2 it is shown that $F_{n+1}^{(0)} = O(h^{q+1})$ only. Using the approximations (2.3.14) and (2.3.15) shows that

$$\underline{\varepsilon}_{n+1}^{(0)} = (q+1)^{-1}\,F_{n+1}^{(0)}\,\underline{c} + O(h^{q+2}) \quad . \tag{2.3.16}$$

Thus, for sufficiently small h $(q+1)^{-1}\,F_{n+1}^{(0)}\,\underline{c}$ is a good approximation to $\underline{\varepsilon}_{n+1}^{(0)}$. Defining $\underline{\ell} = (q+1)^{-1}\,\underline{c}$ we are led to the correction (2.3.1).

The remainder of this section is used to give the proofs of the lemmas and corollaries used to obtain equations (2.3.15) and (2.3.16). Moreover, we shall

derive an expression for $\underline{\varepsilon}_{n+1}$ to show that the order of the predictor-corrector pair is greater than or equal to the order of the predictor, provided the predictor is iterated to convergence.

Lemma 2.3.1

Let $\underline{\varepsilon}^{(0)}_{n+1}$, $\underline{\varepsilon}_n$, and $x^T_{n,i}$, be defined by (2.3.8), (2.3.9), and (2.3.7), respectively. Then

$$\varepsilon^{(0)}_{n+1,i} = \sum_{j=i}^{q} \binom{j}{i} \varepsilon_{n,j} + \sum_{j=q+1}^{\infty} \binom{j}{i} x^T_{n,j} \quad , \quad i=0,1,\ldots,q \; . \tag{2.3.17}$$

Proof: Using the error definition (2.3.8) yields

$$\varepsilon^{(0)}_{n+1,i} = x^T_{n+1,i} - x^{(0)}_{n+1,i} \quad , \quad i=0,1,2,\ldots,q \; .$$

Expanding $x^T_{n+1,i}$ and $x^{(0)}_{n+1,i}$ in Taylor series about the point t_n yields

$$\varepsilon^{(0)}_{n+1,i} = \sum_{j=i}^{q} \binom{j}{i} x^T_{n,j} - \sum_{j=i}^{q} \binom{j}{i} x_{n,j} + \sum_{j=q+1}^{\infty} \binom{j}{i} x^T_{n,j} \tag{2.3.18}$$

which is equivalent to (2.3.17).

Q.E.D.

Corollary 2.3.1

Let $\varepsilon^{(0)}_{n+1,0}$, $\underline{\varepsilon}_n$, and $x^T_{n,i}$, be defined by (2.3.8), (2.3.9), and (2.3.7), respectively. If $\underline{\varepsilon}_n = 0$, then

$$\varepsilon^{(0)}_{n+1,i} = \binom{q+1}{i} x^T_{n,q+1} + O(h^{q+2}) , \quad i=0,1,2,\ldots q.$$

Lemma 2.3.2

Let $F^{(0)}_{n+1}$, $\underline{\varepsilon}_n$, and $x^T_{n,i}$, $i=0,1,2,\ldots$, be defined by (2.3.2), (2.3.9), and (2.3.7), respectively. Then

$$F^{(0)}_{n+1} = \varepsilon^{(0)}_{n+1,1} - h\varepsilon^{(0)}_{n+1,0} E(\xi) , \tag{2.3.20}$$

$$= \sum_{j=0}^{q} (j-hE(\xi)) \varepsilon_{n,q} + \sum_{j=q+1}^{\infty} (j-hE(\xi)) x^T_{n,j} , \tag{2.3.21}$$

where $E(\xi) = f_y(t_{n+1},\xi)$ and ξ is a point between $x^T_{n+1,0}$ and $x^{(0)}_{n+1,0}$.

Proof: Using the definition of $F^{(0)}_{n+1}$, (2.3.2), yields

$$F^{(0)}_{n+1} = hf^{(0)}_{n+1} - x^{(0)}_{n+1,1} .$$

Substituting the first component of the error definition (2.3.8) into $f^{(0)}_{n+1}$ results in

$$F^{(0)}_{n+1} = hf(t_{n+1}, x^T_{n+1,0} - \varepsilon^{(0)}_{n+1,0}) - x^{(0)}_{n+1,1} .$$

Applying the mean value theorem to f, using the argument y, yields

$$F^{(0)}_{n+1} = x^T_{n+1,1} - h\varepsilon^{(0)}_{n+1,0} E(\xi) - x^{(0)}_{n+1,1} . \tag{2.3.22}$$

Using the definition of $\varepsilon^{(0)}_{n+1,1}$, (2.3.8), shows that (2.3.22)

is equivalent to (2.3.20). Using Lemma 2.3.1 to expand (2.3.20) yields (2.3.21).

Q.E.D.

Corollary 2.3.2

Let $F_{n+1}^{(0)}$ and $x_{n,q+1}^{T}$ be defined by (2.3.2) and (2.3.7) respectively. If $\underline{\varepsilon}_n = 0$, then

$$F_{n+1}^{(0)} = (q+1)\, x_{n,q+1}^{T} + O(h^{q+2}) \quad . \qquad (2.3.23)$$

Using Corollaries 2.3.1 and 2.3.2 yields:

Corollary 2.3.3

Let $F_{n+1}^{(0)}$ and $\varepsilon_{n+1,0}^{(0)}$ be defined by (2.3.2) and (2.3.8) respectively. If $\underline{\varepsilon}_n = 0$, then

$$\varepsilon_{n+1,0}^{(0)} = (q+1)^{-1}\, F_{n+1}^{(0)} + O(h^{q+2}) \quad . \qquad (2.3.24)$$

The next lemma, Lemma 2.3.4, is used to show that the order of the predictor-corrector method is greater than or equal to the order of the predictor method, provided that $\ell_1 = 1$. That is, if the error of the predictor is $O(h^{q+1})$, then the error of the predictor-corrector method is at least $O(h^{q+1})$. In particular, we show that

$$F_{n+1} = \frac{\varepsilon^{(0)}_{n+1,1} - hE(\xi_M)\ \varepsilon^{(0)}_{n+1,0}}{1 - h\ell_o E(\xi_M)} \quad , \qquad (2.3.25)$$

where $E(\xi_M) = f_y(t_{n+1}, \xi_M)$ and ξ_M is a point between $x^T_{n+1,0}$ and $x^{(0)}_{n+1,0}$. Using the definition (2.3.9) for $\underline{\varepsilon}_{n+1}$, the implicit form (2.3.5) of the corrector equation (2.3.1) with F_{n+1} replaced by (2.3.25), and the definition (2.3.8) for $\underline{\varepsilon}^{(0)}_{n+1}$, we obtain

$$\underline{\varepsilon}_{n+1} = \underline{\varepsilon}^{(0)}_{n+1} - \left[\frac{\varepsilon^{(0)}_{n+1,1} - hE(\xi_M)\ \varepsilon^{(0)}_{n+1,0}}{1 - h\ell_o E(\xi_M)}\right] \underline{\ell} \qquad (2.3.25)$$

and thus $\underline{\varepsilon}_{n+1}$ is of the same order as $\underline{\varepsilon}^{(0)}_{n+1}$.

The importance of this result is that all the components of $\underline{\ell}$, except ℓ_1, may be chosen so that the truncation error may be lowered or the stability may be improved.

Lemma 2.3.4

Let F_{n+1} and $\underline{\varepsilon}^{(0)}_{n+1}$ be defined by (2.3.6) and (2.3.8) respectively. If $\ell_1 = 1$, and if the corrector is iterated to convergence, then

$$F_{n+1} = \frac{\varepsilon^{(0)}_{n+1,1} - hE(\xi_M)\ \varepsilon^{(0)}_{n+1,0}}{1 - h\ell_o E(\xi_M)} \quad , \qquad (2.3.27)$$

where $E(\xi_M) = f_y(t_{n+1}, \xi_M)$, and ξ_M is a point between $x^T_{n+1,0}$ and $x^{(0)}_{n+1,0}$.

Proof: Using the error definition (2.3.8) of $\underline{\varepsilon}^{(0)}_{n+1}$ and $\ell_1 = 1$ write the first two components of the implicit corrector equation (2.3.5) in the form

$$x_{n+1,0} = x^{(0)}_{n+1,0} + F_{n+1}\,\ell_0 \ ,$$

$$= x^{T}_{n+1,0} - \varepsilon^{(0)}_{n+1,0} + F_{n+1}\,\ell_0 \ , \qquad (2.3.28)$$

and

$$x_{n+1,1} = x^{(0)}_{n+1,1} + F_{n+1} \ ,$$

$$= x^{T}_{n+1,1} - \varepsilon^{(0)}_{n+1,1} + F_{n+1} \ . \qquad (2.3.29)$$

If the corrector equation (2.3.1) is iterated to convergence then

$$hf(t_{n+1}, x_{n+1,0}) - x_{n+1,1} = 0 \ . \qquad (2.3.30)$$

Using (2.3.28) and (2.3.29) to substitute for $x_{n+1,0}$ and $x_{n+1,1}$ (2.3.30) produces

$$hf(t_{n+1}, x^{T}_{n+1,0} - \varepsilon^{(0)}_{n+1,0} + F_{n+1}\,\ell_0) - x^{T}_{n+1,1} + \varepsilon^{(0)}_{n+1,1} - F_{n+1} = 0. \qquad (2.3.31)$$

Applying the mean value theorem to f in (2.3.31) together with the fact that $x^{T}_{n+1,1} = hf(t_{n+1}, x^{T}_{n+1,0})$ yields

$$hF_{n+1}\ell_0 E(\xi_M) - h\varepsilon^{(0)}_{n+1} E(\xi_M) + \varepsilon^{(0)}_{n+1,1} - F_{n+1} = 0 \qquad (2.3.32)$$

which, when solved for F_{n+1}, produces (2.3.27).

Q.E.D.

2.4 Parameter Definitions

In section 2.3, immediately following equation (2.3.16), we choose $\underline{\ell} = (q+1)^{-1}\underline{c}$. Unfortunately, using this choice for $\underline{\ell}$ in the corrector equation (2.3.1) produces an unstable method. However, there exist vectors $\underline{\ell}$ which make the method (2.3.1) stable. One way to determine such an $\underline{\ell}$ is to show that the corrector equation (2.3.1) when iterated to convergence is equivalent to a multistep method and then to impose the stability condition of multistep methods on (2.3.1). We use Osborne's [17] proof to show that the stability of the multivalue method (2.3.5) depends only on $\ell_2, \ell_3, \ldots, \ell_q$.

For the purpose of the analysis, equation (2.3.5) is partitioned into three parts as follows:

$$\begin{pmatrix} x_{n+1,0} \\ x_{n+1,1} \\ \underline{v}_{n+1} \end{pmatrix} = \left(\begin{array}{c|c|c} 1 & 1 & \underline{u}^T \\ \hline 0 & 1 & \underline{d}^T \\ \hline 0 & 0 & B \end{array}\right) \begin{pmatrix} x_{n,0} \\ x_{n,1} \\ \underline{v}_n \end{pmatrix} + F_{n+1} \begin{pmatrix} \ell_0 \\ 1 \\ \underline{k} \end{pmatrix}, \qquad (2.4.1)$$

that is,

$$x_{n+1,0} = x_{n,0} + x_{n,1} + \underline{u}^T\underline{v}_n + \ell_0 F_{n+1}, \qquad (2.4.1a)$$

$$x_{n+1,1} = x_{n,1} + \underline{d}^T\underline{v}_n + F_{n+1}, \qquad (2.4.1b)$$

$$\underline{v}_{n+1} = B\underline{v}_n + F_{n+1}\,\underline{k}, \qquad (2.4.1c)$$

where

$$\underline{v}_n = \mathrm{col}(x_{n,2}, x_{n,3}, \ldots, x_{n,q}) \ , \qquad (2.4.2)$$

$$\underline{u} = \mathrm{col}(1, 1, 1, \ldots, 1) \ , \qquad (2.4.3)$$

$$\underline{d} = \mathrm{col}(2, 3, \ldots, q) \ , \qquad (2.4.4)$$

$$\underline{k} = \mathrm{col}(\ell_2, \ell_3, \ldots, \ell_q) \ , \qquad (2.4.5)$$

and

$$B = \begin{pmatrix} \binom{2}{2} & \binom{3}{2} & \binom{4}{2} & \cdots & \binom{q}{2} \\ & \binom{3}{3} & \binom{4}{3} & \cdots & \binom{q}{3} \\ & & \binom{4}{4} & \cdots & \binom{q}{4} \\ 0 & & & \ddots & \\ & & & & \binom{q}{q} \end{pmatrix} \qquad (2.4.6)$$

To determine a particular $\underline{k}$ we use the following two lemmas, whose proofs are deferred to the end of this section, to show that (2.4.1) is equivalent to a multistep method whose parameters depend on $\underline{k}$.

Lemma 2.4.1

Let the system of equations (2.4.1) be given. Let

$$G_n = h \sum_{j=1}^{q} \eta_j (f_{n+j-1} + \ell_0 \nabla f_{n+j}) \ , \qquad (2.4.7)$$

and

$$H_n = (\underline{u} - \ell_0 \underline{d})^\tau \sum_{j=1}^{q} \eta_j \underline{v}_{n+j-1} \quad , \tag{2.4.8}$$

where the η_i's are arbitrarily chosen constants. Then

$$\sum_{j=1}^{q} \eta_j \nabla Y_{n+j} = G_n + H_n \quad . \tag{2.4.9}$$

Lemma 2.4.2

Let the system of equation (2.4.1) be given. Then

$$\underline{v}_{n+i} = A^i \underline{v}_n + h\left(\sum_{j=0}^{i-1} \nabla f_{n+i-j} A^j\right)\underline{k} \quad , \tag{2.4.10}$$

where $A = B - \underline{k}\,\underline{d}^\tau$.

Substituting (2.4.10) in (2.4.8) yields

$$H_n = I_n + J_n \quad , \tag{2.4.11}$$

where

$$I_n = (\underline{u} - \ell_0 \underline{d})^\tau h\left(\sum_{i=1}^{q} \eta_i \sum_{j=0}^{i-1} \nabla f_{n+i-j-1} A^j\right)\underline{k} \quad , \tag{2.4.12}$$

and

$$J_n = (\underline{u} - \ell_0 \underline{d})^\tau \left(\sum_{j=1}^{q} \eta_j A^{j-1}\right)\underline{v}_n \quad . \tag{2.4.13}$$

Substituting $I_n + J_n$ for H_n into (2.4.9) and examining the resulting equation shows that $\sum \eta_j \nabla Y_{n+j}$ is a

linear combination of all the Y values present in the equation. G_n and I_n are linear combinations of f values. Moreover, if J_n can be made identically zero, then the resulting equation

$$\sum_{j=1}^{q} \eta_j \nabla Y_{n+j} = G_n + I_n \qquad (2.4.14)$$

is a multistep method. One way to make J_n equal zero is to choose the η_i's to be the coefficients of the characteristic polynomial of A. With this choice the Hamilton-Caley theorem asserts

$$\sum_{j=1}^{q} \eta_j A^{j-1} \equiv 0 \quad . \qquad (2.4.15)$$

Thus, (2.4.13) becomes identically zero and (2.4.9) becomes the multistep method (2.4.14).

We note that equation (2.4.14) can be written

$$(E-1)\psi(E)Y_n = G_n + I_n \quad , \qquad (2.4.16)$$

where $E^j Y_n = Y_{n+j}$, and $\psi(E)$ is a polynomial of degree (q-1) defined by

$$\psi(z) = \sum_{j=1}^{q} \eta_j z^{j-1} \quad . \qquad (2.4.17)$$

We recall, ref. Henrici [6], that a multistep method can be written in the form

$$\alpha(E)Y_n = h\beta(E)f_n \qquad (2.4.18)$$

where α and β are polynomials. Moreover, we recall that given both the α-polynomial of a multistep method and its order, the β polynomial is determined. Furthermore, a multistep method is said to be stable if all roots of the α-polynomial are less than or equal to one in magnitude and if those roots that are equal to one are simple.

Evidently, comparing equation (2.4.16) with the general form of multistep methods (2.4.18) shows that

$$\alpha(z) = (z - 1)\psi(z) \quad . \qquad (2.4.19)$$

Clearly, this polynomial has one root equal to one and its remaining roots coinciding with the roots of the polynomial $\psi(z)$ defined by (2.4.17). Now, recall that $\psi(z)$ is the characteristic equation of $A = B - \underline{k}\,\underline{d}^T$. Moreover, recall that A has been obtained from $Q - \underline{\ell}(\underline{\delta}_1^T Q)$ by deleting the first two rows and columns. With these remarks we use the following theorem stated and proved by Gear [5] to assert the existence of a unique $\underline{k}$.

Theorem

If Q is the $(q+1)\times(q+1)$ Pascal Matrix defined by (2.2.9) and $\underline{\delta}_i$ is a vector with a one in the i-th position and zeros elsewhere, then for any set of q-i+1

numbers $\{\lambda_i\}$ a column vector $\underline{\ell}$ exists such that the matrix

$$(I - \underline{\ell}\,\underline{\delta}_i^{\tau})^M Q$$

has i eigenvalues equal to one and q-i+1 eigenvalues equal to the set $\{\lambda_i\}$. The eigenvalues are independent of the first i elements of $\underline{\ell}$ and uniquely determine the last q-i+1 elements of $\underline{\ell}$.

In the remainder of this study we assume that $\lambda_i = 0$, i=1,2,...,q-1, where $\{\lambda_i\}$ is the set of roots of $\psi(z)$. Consequently,

$$\alpha(z) = (z - 1)\, z^{q-1}\,, \qquad (2.4.20)$$

$A^{q-1} = 0$, $\eta_q = 1$, and $\eta_i = 0$, for i=1,2,...,q-1. Moreover, with this choice of $\{\lambda_i\}$ Nordsieck's method is stable. Furthermore, in the next section it is shown that ℓ_0 can be chosen so that the local truncation error of the (q+1) value method is $O(h^{q+2})$. Hence, from previously made remarks it follows that Nordsieck's method is equivalent to the Adams-Moulton method.

The proofs of the lemmas complete this section.

Proof of Lemma 2.4.1

Write equations (2.4.1a) and (2.4.1b) in the form

$$Y_{n+1} = Y_n + hf_n + \underline{u}^{\tau}\underline{v}_n + \ell_0 F_{n+1} \qquad (2.4.21)$$

and

$$F_{n+1} = h\nabla f_{n+1} - \underline{d}^T \underline{v}_n \quad . \tag{2.4.22}$$

Substitute (2.4.22) for F_{n+1} into (2.4.21) and add j to the subscripts to obtain

$$Y_{n+j+1} = Y_{n+j} + hf_{n+j} + (\underline{u} - \ell_0 \underline{d})^T \underline{v}_{n+j} + \ell_0 h\nabla f_{n+j+1} \quad . \tag{2.4.23}$$

Use the coefficients η_j to form the linear combination

$$\sum_{j=1}^{q} \eta_j Y_{n+j} = \sum_{j=1}^{q} \eta_j (Y_{n+j-1} + hf_{n+j-1} + (\underline{u} - \ell_0 \underline{d})^T \underline{v}_{n+j-1} + \ell_0 h\nabla f_{n+j}) \quad . \tag{2.4.24}$$

Recall the definitions (2.4.7) and (2.4.8) of G_n and H_n respectively and write (2.4.24) in the form (2.4.9) to complete the proof.

Q.E.D.

Proof of Lemma 2.4.2

Substitute equation (2.4.2b) in the form (2.4.22) for F_{n+1} into equation (2.4.1c) to obtain

$$\underline{v}_{n+1} = B\underline{v}_n + (h\nabla f_{n+1} - \underline{d}^T \underline{v}_n)\, \underline{k} \quad . \tag{2.4.25}$$

Equation (2.4.25) is a difference equation in standard form which can be solved to obtain (2.4.10).

Q.E.D.

2.5 Truncation Error

The first component ℓ_0 of $\underline{\ell}$ is used to minimize the truncation error.

The local truncation error T_n of the method (2.4.1) is defined by

$$y_{n+1} = y_n + hf_n^T + (\underline{u} - \ell_0\underline{d})^\tau v_n^T + \ell_0 h\nabla f_{n+1}^T + T_n \quad , \qquad (2.5.1)$$

where $\underline{v}_n^T$ is the vector calculated from (2.4.10) by assuming that past f values are exact and that $f_n^T = f(t,y_n)$.

The following lemmas express T_n in terms of the true solution.

Lemma 2.5.1

The vector $\underline{v}_n^T$ of equation (2.5.1) is given by

$$\underline{v}_n^T = \sum_{i=1}^{\infty} i\, x_{n,i}^T\, \underline{\sigma}_i \quad , \qquad (2.5.2)$$

where

$$\underline{\sigma}_i = (-1)^i \sum_{j=0}^{q-2} ((j+1)^{i-1} - j^{i-1})A^j\underline{k} \quad . \qquad (2.5.3)$$

Lemma 2.5.2

Given relation (2.5.2), equation (2.5.1) can be written

$$y_{n+1} = y_n + hf_n^T + \sum_{i=2}^{\infty} \theta_i\, x_{n,i}^T + T_n \quad , \qquad (2.5.4)$$

where

$$\theta_i = i(\ell_0 + (\underline{u} - \ell_0\underline{d})^T \underline{\sigma}_i) \quad ,$$

$$= i(\ell_0(1 - \underline{d}^T\underline{\sigma}_i) + \underline{u}^T\underline{\sigma}_i) \quad . \qquad (2.5.5)$$

Since the error in Y_{n+1} is at least $O(h^{q+1})$ each θ_i in (2.5.4) must equal one for $i \le q$. Setting

$$\ell_0 = (\frac{1}{q+1} - \underline{u}^T\underline{\sigma}_{q+1})(1 - \underline{d}^T\underline{\sigma}_{q+1})^{-1} \qquad (2.5.6)$$

produces $\theta_{q+1} = 1$. Thus, the local truncation error in the first component of the corrector (2.4.1) becomes $O(h^{q+2})$.

The proofs of the lemmas complete this section.

Proof of Lemma 2.5.1

Obtain $\underline{v}_n^T$ from (2.4.10) by using ∇f^T in place of ∇f evaluated at the points t_n, t_{n-1}, ..., $t_{n-(q-2)}$. That is,

$$\underline{v}_n^T = h \sum_{j=0}^{q-2} \nabla f_{n-j}^T A^j \underline{k} \quad . \qquad (2.5.7)$$

Expand each ∇f_{n-j}^T in (2.5.7) in a Taylor series about the point t_n to obtain

$$\underline{v}_n^T = h \sum_{j=0}^{q-2} (f_{n-j}^T - f_{n-(j+1)}^T) A^j \underline{k} \quad ,$$

$$= \sum_{j=0}^{q-2} \sum_{i=2}^{\infty} (-1)^i ((j+1)^{i-1} - j^{i-1}) \, i \, x_{n,i}^T A^j \underline{k} \quad ,$$

$$= \sum_{i=2}^{\infty} (-1)^i \, i \left(\sum_{j=0}^{q-2} ((j+1)^{i-1} - j^{i-1}) A^j \underline{k} \right) x_{n,i}^T \qquad (2.5.8)$$

Using the definition (2.5.3) of $\underline{\sigma}_i$ reduces (2.5.8) to (2.5.2).

Q.E.D.

Proof of Lemma 2.5.2

Expand ∇f_{n+1}^T in a Taylor series about the point t_n to produce

$$h \nabla f_{n+1}^T = \sum_{i=2}^{\infty} i \, x_{n,i}^T \qquad (2.5.9)$$

Substitute (2.5.2) and (2.5.9) for $\underline{v}_n^T$ and $h\nabla f_{n+1}^T$ in (2.5.1) to produce (2.5.4).

Q.E.D.

CHAPTER III

STEPSIZE AND ORDER

3.1 Introduction

In this chapter we outline the technique for changing the stepsize and order, examine the error introduced by stepchanges, and outline an algorithm to be applied when the stepsize is reduced.

Suppose that at the point t_n the stepsize h is to be changed to $h^* = \rho h$. Then, the value of the i-th component of $\underline{x}_n$ must be transformed from $h^i y_n^{(i)}/i!$ to $h^{*i} y_n^{(i)}/i!$. In order to distinguish between these two quantities we denote the latter by $x^*_{n,i}$. Consequently, we write $\underline{x}^*_n = \text{col}(x^*_{n,0}, x^*_{n,1}, \ldots, x^*_{n,q})$ and $t^*_n = t_n + h^*$.

The transformation required to carry $\underline{x}_n$ into $\underline{x}^*_n$ is easily determined by using the vector $\underline{z}_n$ defined by (2.2.2) and the matrix D(h) defined by (2.2.10). From equations (2.2.18) and (2.2.19) it follows then that

$$\underline{z}_n = D^{-1}(h)\, \underline{x}_n \ ,$$

and

$$\underline{x}^*_n = D(h^*)\, \underline{z}_n \ .$$

Hence,

$$\underline{x}^*_n = R(\rho)\underline{x}_n \ , \qquad (3.1.1)$$

where

$$R(\rho) = D(h^*)\, D^{-1}(h) ,$$

which is equivalent to

$$R(\rho)_{ij} = \rho^{i-1}\, \delta_{ij} , \qquad (3.1.2)$$

$i,j = 0,1,2,\ldots,q.$

3.2 Stepsize-Changing Errors

Empirical evidence shows that applying the transformation $R(\rho)$ defined by (3.1.2) to $\underline{x}_n$ to change the stepsize is a good way to increase the stepsize; but substantial errors are introduced when $R(\rho)$ is used to decrease the stepsize. These errors are introduced because of the way the interpolation polynomial is used. Before examining the interpolation polynomial in general consider Example 3.2.1.

The last column of Table 3.2.1 shows the error as the difference between $y^{(i)}(t)$ and $P_{1,1}^{(i)}(t)$ for $i=0,1,2,3$, where $P_{1,1}(t)$ is the polynomial defined by $\underline{x}_1$ at the point $t = 1.1$. These errors are small for $i = 0$ and $i = 1$, however, they grow rapidly for i larger than one. These large errors in the higher derivatives of the approximate solution indicate that the solution polynomial oscillates about the true solution.

Example 3.2.1

Integrate the initial value problem

$$y^{(1)} = 5t^4, \quad y(1) = 1 , \tag{3.2.3}$$

one step forward with h = 0.1, q = 3, $\underline{\ell}$ = col(3/8, 1, 3/4, 1/6). Calculating the derivatives of the true solution $y(t) = t^5$ yields the starting vector

$$\underline{x}_0 = \text{col}(1.0,\ 0.5,\ 0.1,\ 0.01) \tag{3.2.4}$$

which defines the starting polynomial

$$P_0(t) = 1.0 + 5.0(t-1) + \frac{20.0}{2!}(t-1)^2 + \frac{60.0}{3!}(t-1)^3 .$$

The predicted solution vector $\underline{x}_1^{(0)}$ = col(1.61, 0.73, 0.13, 0.01) defines the polynomial

$$P_{1,0}(t) = 1.61 + 7.3(t-1.1) + \frac{26.0}{2!}(t-1.1)^2 + \frac{60.0}{3!}(t-1.1)^3 .$$

The corrected solution vector $\underline{x}_1^{(1)}$, given in the third column of Table 2.3.1, defines the polynomial

$$P_{1,1}(t) = 1.6105125 + 7.3205(t-1.1) + \frac{26.3075}{2!}(t-1.1)^2 + \frac{62.05}{3!}(t-1.1)^3 .$$

Values of the Polynomials (3.2.3) and (3.2.4)

Derivative	$P_{1,0}(1.1)$	$P_{1,1}(1.1)$	y(1.1)	Error
0-th	1.61	1.6105125	1.61051	-0.00025875
1st	7.3	7.3205	7.3205	0.0
2nd	26.0	26.3075	26.62	0.3125
3rd	60.0	62.05	72.6	10.55

Table 3.2.1

We shall now examine how the interpolation polynomial in Nordsieck's method is used when the stepsize is changed. We derive a transformation S which takes the vector $\underline{x}_n$ into another vector $\underline{p}_n$ which represents the interpolation polynomial P(t) in terms of past values. The transformation S and the vector $\underline{p}_n$ are used to transform Nordsieck's formula into an equivalent formula using past values of P(t). We then show how the oscillations present in P(t) are translated into the scaled derivatives. It can then be seen how step-changes introduce errors into the solution vector $\underline{x}_n$.

Recall that $x_{n,j}$ consists of the j-th scaled derivative of P(t). Expand $P^{(1)}(t - ih)$ in a Taylor series about the point t_n to obtain

$$P^{(1)}(t_n - ih) = \sum_{j=0}^{q-1} (-i)^j \frac{h^j}{j!} P^{(j+1)}(t_n) ,$$

$$= h^{-1} \sum_{j=1}^{q} (-i)^{j+1} j\, x_{n,j} , \quad i=0,1,2,\ldots,q-1 . \tag{3.2.5}$$

Defining

$$\underline{p}_n = \text{col}(P(t_n), hp^{(1)}(t_n), \ldots, hp^{(1)}(t_n - (q-1)h))$$

allows us to write

$$\underline{p}_n = S^{-1} \underline{x}_n , \tag{3.2.6}$$

where

$$S^{-1}_{ij} = \begin{cases} 1, & \text{if } i=0 \text{ and } j=0 , \\ 0, & \text{if } i=0 \text{ and } j>0 , \\ 0, & \text{if } j=0 \text{ and } i<0 , \\ (-1)^{j+1} j , & \text{otherwise} . \end{cases}$$

It is not hard to show that S^{-1} is non-singular. Hence, for example, when $q = 4$,

$$S = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 11/12 & -3/2 & 3/4 & -1/6 \\ 0 & 1/3 & -5/6 & 2/3 & -1/6 \\ 0 & 1/24 & -1/8 & 1/8 & -1/24 \end{pmatrix} .$$

To find the equivalent of Nordsieck's formula in terms of past values write equation (2.4.1b) in the form

$$F_{n+1} = hf_{n+1} - (Q\underline{x}_n)_1 . \tag{3.2.7}$$

Substitute (3.2.7) for F_{n+1} into the implicit corrector equation (2.3.5) to produce

$$\underline{x}_{n+1} = (Q - \underline{\ell}(\underline{\delta}_1^T Q))\underline{x}_n + hf_{n+1}\,\underline{\ell} \quad . \tag{3.2.8}$$

Substitute (3.2.6) for $\underline{x}_{n+1}$ and $\underline{x}_n$ in (3.2.8) and obtain

$$\underline{p}_{n+1} = S^{-1}(Q - \underline{\ell}(\underline{\delta}_1^T Q))\, S\underline{p}_n + hf_{n+1}S^{-1}\underline{\ell} \quad . \tag{3.2.9}$$

We note that both methods (3.2.8) and (3.2.9) will produce the same solution to a given initial value problem if:

(i) No starting errors are committed.

(ii) A constant stepsize is used.

(iii) No roundoff error is introduced.

Suppose now that at the point t_n the stepsize h is increased to 2h. Retaining conditions (i) and (iii) above, $\underline{x}_n$ is then replaced by $\underline{x}_n^*$, where

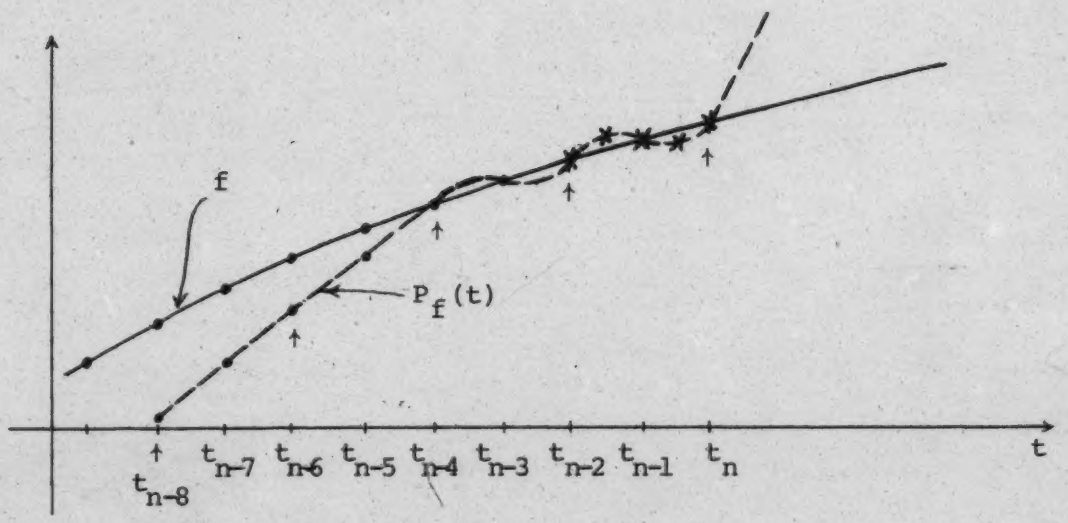
$$\begin{aligned} \underline{x}_n^* &= R(2)\ \underline{x}_n \ , \\ &= R(2)\ S\ \underline{p}_n \ , \\ &= S\ \underline{p}_n^* \ , \end{aligned} \qquad (3.2.10)$$

and

$$\underline{p}_n^* = \mathrm{col}(P(t_n), 2hP^{(1)}(t_n), 2hP^{(1)}(t_n-2h), \ldots, 2hP^{(1)}(t_n-(q-1)2h)) \ .$$

Relation (3.2.10) shows that Nordsieck's method does not use values of f to continue the integration, rather, it uses values calculated from the interpolating polynomial P(t) at points spaced a distance 2h apart. These points are marked by (†) in Graph 3.2.1. Examining Graph 3.2.1 more closely indicates that the errors $\varepsilon_{n-i} = f_{n-i} - P^{(1)}(t_{n-i})$ are zero for i=0,2,4 and are non-zero and of the same sign for i=6,8. Moreover, these errors ε_{n-i} increase in magnitude as i increased. However, premultiplying $\underline{p}_n^*$ by the matrix S to form the vector $\underline{x}_n^*$ causes, due to the alternating signs of the row entries of S, these errors partly to cancel. In addition, the row entries S_{ij} strongly decrease in

Graph of $P_f(t)$ and f versus t



× points sampled when h is halved.

↑ points sampled when h is doubled.

GRAPH 3.2.1

magnitude with increasing j when $j \geq i$. This reduces the contribution of the large errors. Consequently, no difficulties arise when the stepsize is doubled. Clearly, a similar illustration and conclusion can be given for increasing the stepsize to an arbitrary h* using a method of order q.

Suppose now that the stepsize is halved from h to h/2. Then, (3.2.10) becomes

$$\underline{x}_n^* = S\, \underline{p}_n^* \tag{3.2.11}$$

where

$$\underline{p}_n^* = \text{col}\left(P(t_n),\ \frac{h}{2} P^{(1)}(t_n),\ \frac{h}{2} P^{(1)}(t_n - \frac{h}{2}),\ \ldots,\ \frac{h}{2} P^{(1)}(t_n - (q-2)\ \frac{h}{2})\right) .$$

In this case values of the interpolating polynomial P(t) spaced a distance h/2 apart are used. These values are marked (x) in Graph 3.2.1. It can now be seen from the graph that some of the errors which are close to the point t_n are non-zero. These errors are then multiplied by the large entries of S. Thus, relatively large errors in $\underline{x}_n^*$ result. Consequently, care must be exercised when reducing the stepsize.

3.3 Tests for Stepsize and Order

We recall that in section 2.5 the parameter ℓ_0 is determined so that the (q+1)-value multivalue method

has a local truncation error $O(h^{q+2})$. Unfortunately, if the error is $O(h^{q+2})$ then good estimates of the local truncation errors are difficult to obtain. However, if the error is $O(h^{q+1})$ then good estimates are obtainable.

After experimenting, the following method of choosing $\underline{\ell}$ is used. Let $\underline{\ell}_q$ be the vector which makes the multivalue method (2.3.1) equivalent to the q-th order Adams-Moulton method. Our program uses vectors $\underline{\tilde{\ell}}_q$ defined by

$$\tilde{\ell}_{q,j} = \begin{cases} \ell_{q-1,j} , & \text{if } j = 0 , \\ \ell_{q,j} , & \text{if } j > 0 . \end{cases}$$

This choice insures good stability characteristics and produces local truncation errors $O(h^{q+1})$.

To see why it is desirable not to use maximum order (q+1)-value methods suppose that such a method is used. The truncation error of the method is $O(h^{q+2})$. Then, truncation error estimates for the (q-1), q, and (q+1)- value methods are required. These estimates are of the form $C_{q+1}\, y_{n+1}^{(q+1)}\, h^{q+1}$, $C_{q+2}\, y_{n+1}^{(q+2)} h^{q+2}$, and $C_{q+3}\, y_{n+1}^{(q+3)}\, h^{q+3}$, respectively, where C_{q+1}, C_{q+2}, and C_{q+3}, are known constants. Hence, we require estimates of $y_{n+1}^{(q+1)}$, $y_{n+1}^{(q+2)}$, and $y_{n+1}^{(q+3)}$. Examining (2.4.1) shows that the last component of that system of equations can be written in the form

$$\nabla x_{n+1,q} = \ell_q F_{n+1} \quad . \tag{3.3.1}$$

Noting that

$$\nabla x_{n+1,q} = \frac{h^q}{q!} y_{n+1}^{(q)} - \frac{h^q}{q!} y_n^{(q)} \quad ,$$

$$\approx \frac{h^{q+1}}{q!} y_n^{(q+1)} \quad ,$$

provides an estimate of $y_{n+1}^{(q+1)} h^{q+1}$. Moreover, the first and second backward differences of $\ell_q F_{n+1}$ can then be used to estimate $y_{n+1}^{(q+2)} h^{q+2}$ and $y_{n+1}^{(q+3)} h^{q+3}$ respectively. However, in practice we found that $\ell_q F_{n+1}$ approximates $y_{n+1}^{(q+1)} h^{q+1}/q!$ with only one significant figure accuracy. Consequently, backward differences of $\ell_q F_{n+1}$ are not accurate with any signifcant figures. Therefore, two of the required estimates are not accurate.

On the other hand, if the q-value method has truncation error $O(h^{q+1})$ then the truncation errors take the form $C_q y_{n+1}^{(q)} h^q$, $C_{q+1} y_{n+1}^{(q+1)} h^{q+1}$, and $C_{q+2} y_{n+1}^{(q+2)} h^{q+2}$. These errors are now estimated by

$$T_{q-1,EST} = C_q q! x_{n+1,q} \quad , \tag{3.3.2}$$

$$T_{q,EST} = C_{q+1} q! \ell_q F_{n+1} \quad , \tag{3.3.3}$$

$$T_{q+1,EST} = C_{q+2} q! \ell_q \nabla F_{n+1} \quad , \tag{3.3.4}$$

where C_q, C_{q+1}, and C_{q+2}, are known constants. In practice, we found that $x_{n+1,q}$ and $\ell_q F_{n+1}$ produce estimates which are accurate to at least one significant figure. Thus, with this choice of the parameter ℓ_0 two of the required three estimates are reasonably accurate.

Using the estimates (3.3.2), (3.3.3), and (3.3.4), of local truncation errors we calculate three stepsize estimates h^*_{q-1}, h^*_q, and h^*_{q+1}, respectively, so that using these stepsizes and formulae of corresponding order produces local truncation error estimates which are equal to a given absolute local truncation error tolerance τ_n.

To calculate h^*_{q-1} we use the fact that the local truncation error for the method of order (q-1) has the form $T_{q-1} = C_q h^q y^{(q)}_{n+1}$. The last component $x_{n+1,q}$ of $\underline{x}_{n+1}$ provides the estimate (3.3.2) of T_{q-1}. To introduce h^*_{q-1} into $T_{q-1,EST}$ and to make $T_{q-1,EST}$ equal to τ_n set

$$(h^*_{q-1}/h)^q \, C_q \, q! \, x_{n+1,q} = \tau_n$$

which can be written in the form

$$h^*_{q-1} = \left(\frac{\tau_n}{C_q \, q! \, x_{n+1,q}} \right)^{1/q} h \quad . \qquad (3.3.5)$$

The quantity h^*_{q-1} is now an estimate of the stepsize which would have produced a truncation error estimate equal to τ_n had the formula of order (q-1) been used.

In exactly the same way (3.3.3) and (3.3.4) are used to calculate the remaining estimates

$$h^*_q = \left(\frac{\tau_n}{C_{q+1}\ q!\ \ell_q\ F_{n+1}}\right)^{1/q+1} h \ , \qquad (3.3.6)$$

and

$$h^*_{q+1} = \left(\frac{\tau_n}{C_{q+2}\ q!\ \ell_q\ F_{n+1}}\right)^{1/q+2} h \ . \qquad (3.3.7)$$

In practice, these choices tend to cause too much stepsize changing. Thus, when changing stepsize the estimated stepsize should be reduced by a factor $\alpha < 1$. The program in the Appendix uses $\alpha = 0.9$.

We must now decide which h^*_i to use. The algorithm which follows shows how the program in the Appendix chooses the stepsize and order.

(i) Specify a set of bias factors β_{q-1}, β_q and β_{q+1}, which must be chosen so as to reflect the amount of work expended when using order q-1, q, q+1, formulae respectively.

(ii) Using h for the stepsize compute the next solution point.

(iii) Calculate $\beta_q\ h^*_q$, if $0.9\ h \le \beta_q\ h^*_q \le 1.1\ h$ do (viii) next.

(iv) Calculate $\beta_{q-1}\, h^*_{q-1}$ and $\beta_{q+1}\, h^*_{q+1}$, find the number q^* so that

$$\beta_{q^*}\, h^*_{q^*} = \max(\beta_{q-1}\, h^*_{q-1},\ \beta_q\, h^*_q,\ \beta_{q+1}\, h^*_{q+1}),$$

set $h^* = \alpha\beta_{q^*}\, h^*_{q^*}$, and the order to q^*.

(v) If $h^* \geq h$ do (vii) next. If the stepsize is reduced four times on a given step reduce the order to 2.

(vi) Repeat the step with stepsize h^* and return to (iii).

(vii) Set the stepsize to h^*.

(viii) Accept the step and return to (ii).

3.4 A Stepsize Changing Algorithm

In this section, following a suggestion by Krough [14], we develop the correction

$$\underline{x}_n^{(c)} = \underline{x}_n + F_{n+1}^{(0)}\, \underline{c}^* \quad , \tag{3.4.1}$$

where

$$c_i^* = \frac{Q_{i,q+1} - (q+1)\ell_i}{\sum_{j=1}^{q} Q_{j,q+1} - (q+1)(\underline{\ell})_1} \quad , \quad i=0,1,2,\ldots,q \ ,$$

to be applied to $\underline{x}_n$ immediately before changing the stepsize from h to h^*.

Note that before changing the stepsize to h* a tentative step is made with stepsize h, therefore, $F_{n+1}^{(0)}$ is available without extra function evaluations.

In our analysis we assume that

(i) The exact solution has been obtained at the points $t_0, t_1, \ldots, t_{n-1}$.

(ii) The solution is a (q+1) degree polynomial.

(iii) A q-th order Nordsieck method is used with one application of the corrector formula.

In addition, the predictor matrix will have dimension q+2.

From assumptions (i) and (ii) it follows that

$$\underline{x}_n^T = Q\, \underline{x}_{n-1}^T \quad , \tag{3.4.2}$$

and

$$\underline{x}_{n+1}^T = Q\, \underline{x}_n^T \quad , \tag{3.4.3}$$

where

$$\underline{x}_{n-1}^T = \mathrm{col}(x_{n-1,0}^T, x_{n-1,1}^T, \ldots, x_{n-1,q+1}^T) \, .$$

Because Q is of dimension q+2 we augment the vector $\underline{x}_{n-1}$ with a zero. Thus,

$$\underline{x}_{n-1} = \mathrm{col}(x_{n-1,0}, x_{n-1,1}, \ldots, x_{n-1,q}, 0) \, .$$

Furthermore, we augment $\underline{\ell}$ with a zero, so that,

$$\underline{\ell} = \text{col}(\ell_0, \ell_1, \ldots, \ell_q, 0) \ .$$

Making two steps with Nordsieck's method yields

$$\underline{x}_n^{(0)} = Q\, \underline{x}_{n-1} \quad , \tag{3.4.4}$$

$$\underline{x}_n = Q\, \underline{x}_{n-1} + F_n^{(0)}\, \underline{\ell} \quad , \tag{3.4.5}$$

$$\begin{aligned} \underline{x}_{n+1}^{(0)} &= Q\, \underline{x}_n \quad , \\ &= Q^2 \underline{x}_{n-1} + F_n^{(0)}\, Q\, \underline{\ell} \quad , \end{aligned} \tag{3.4.6}$$

$$\underline{x}_{n+1} = \underline{x}_{n+1}^{(0)} + F_{n+1}^{(0)}\, \underline{\ell} \quad . \tag{3.4.7}$$

The errors in (3.4.4) to (3.4.7) are given by

$$\begin{aligned} \underline{\varepsilon}_n^{(0)} &= \underline{x}_n^T - \underline{x}_n^{(0)} \quad , \\ &= Q(\underline{x}_{n-1}^T - \underline{x}_{n-1}) \quad , \end{aligned} \tag{3.4.8}$$

$$\begin{aligned} \underline{\varepsilon}_n &= \underline{x}_n^T - \underline{x}_n \quad , \\ &= \underline{\varepsilon}_n^{(0)} - F_n^{(0)}\, \underline{\ell} \quad , \end{aligned} \tag{3.4.9}$$

$$\underline{\varepsilon}_{n+1}^{(0)} = Q^2 (\underline{x}_{n-1}^T - \underline{x}_{n-1}) - F_n^{(0)}\, Q\, \underline{\ell} \ , \tag{3.4.10}$$

$$\underline{\varepsilon}_{n+1} = \underline{\varepsilon}_{n+1}^{(0)} - F_{n+1}^{(0)}\, \underline{\ell} \quad . \tag{3.4.11}$$

In order to develop the correction procedure (3.4.1) we analyze equations (3.4.8)-(3.4.11) and use

the lemmas of section 3.2. Using equation (3.4.8) and (3.4.9) yields

$$\varepsilon_{n,i} = Q_{i,q+1}\, x^T_{n-1,q+1} - F^{(0)}_n\, \ell_i \ , \qquad (3.4.12)$$

for i=0,1,2,...,q. From Corollary 2.3.2 follows that

$$F^{(0)}_n = (q+1)\, x^T_{n-1,q+1} \ . \qquad (3.4.13)$$

Thus,

$$\varepsilon_{n,i} = (Q_{i,q+1} - (q+1)\, \ell_i)\, x^T_{n-1,q+1} \ . \qquad (3.4.14)$$

Using Lemma (2.3.2) yields

$$F^{(0)}_{n+1} = \varepsilon^{(0)}_{n+1,1} + O(h^{q+2}) \ . \qquad (3.4.15)$$

If assumption (ii) above is dropped, equations (3.4.12) and (3.4.13) become

$$\varepsilon_{n,i} = Q_{i,q+1}\, x^T_{n-1,q+1} - F^{(0)}_n\, \ell_i + O(h^{q+2}) \ , \qquad (3.4.16)$$

and

$$F^{(0)}_n = (q+1)\, x^T_{n-1,q+1} + O(h^{q+2}) \ . \qquad (3.4.17)$$

Equation (3.4.10) yields

$$\varepsilon^{(0)}_{n+1,1} = \Big(\sum_{j=1}^{q} j\, Q_{j,q+1}\Big)\, x^T_{n-1,q+1} - F^{(0)}_n\, (Q\underline{\ell})_1 \ . \qquad (3.4.18)$$

Thus,

$$F_{n+1}^{(0)} = \left(\sum_{j=1}^{q} j Q_{j,q+1} - (q+1)(Q\underline{\ell})_1 \right) x_{n-1,q+1}^T + O(h^{q+2}) \quad . \tag{3.4.19}$$

Defining c_i^* by (3.4.1) and using equations (3.4.14) and (3.4.19) shows that

$$\underline{\varepsilon}_n - F_{n+1}^{(0)} \underline{c}^* = O(h^{q+2}) \quad . \tag{3.4.20}$$

That is,

$$\underline{x}_n^T - (\underline{x}_n + F_{n+1}^{(0)} \underline{c}^*) = O(h^{q+2}) \quad . \tag{3.4.21}$$

Thus, by adding the correction $\underline{c}^* F_{n+1}^{(0)}$ the error in $\underline{x}_n$ is decreased by a factor of h.

CHAPTER IV

OUTLINE OF THE COMPUTER PROGRAM

4.1 Introduction

A computer program, called NDSK, has been written to solve the system of ordinary differential equations

$$y_i^{(1)} = f_i(t;y_1,y_2,\ldots,y_I), \quad i=1,2,\ldots,I, \quad (4.1.1)$$

$$y_i(t_0) = y_{i,0} \ ,$$

where $1 \le I \le 10$, and $t_0 \le t \le t_f$.

NDSK uses Nordsieck's method to generate a sequence of solution points $\underline{Y}(t_1)$, $\underline{Y}(t_2)$, ..., $\underline{Y}(t_f)$ on the basis of parameters specified by the user. At each step the stepsize and the order are determined by the program; but, provisions are made to enable the user to override the programs choice of the stepsize and to limit the order. This ability to vary the order allows NDSK to handle mild discontinuities in the driving function f and its derivatives.

A general description of NDSK is provided in section 4.2 and details are given by comment statements in the program listing. Section 4.3 provides the details of NDSK's calling sequence, and section 4.4 gives the calling sequence of the subroutine FEVAL which evaluates

the driving function. Section 4.5 provides some detail on how the stepsize and the order are chosen.

Since the present version of the program is the product of considerable experimentation, where we have tried several innovations that appeared promising, we conclude this chapter with a brief outline of its development.

4.2 General Description of the Program

The user's calling program communicates with SUBROUTINE NDSK which organizes and supervises the integration process by calling several subroutines that perform various functions such as:

(i) calculating the next solution point,

(ii) testing for stepsize and order,

(iii) defining constants and coefficients,

(iv) printing the solution.

The interaction of these functions is indicated in Figure 4.2.1.

When control passes to NDSK the input arguments are checked for errors. If any errors are discovered these are corrected, if possible. The arguments are then transferred into the working array A(10,18) located in COMMON BLK1 which is available to all subroutines.

FLOWCHART FOR NDSK

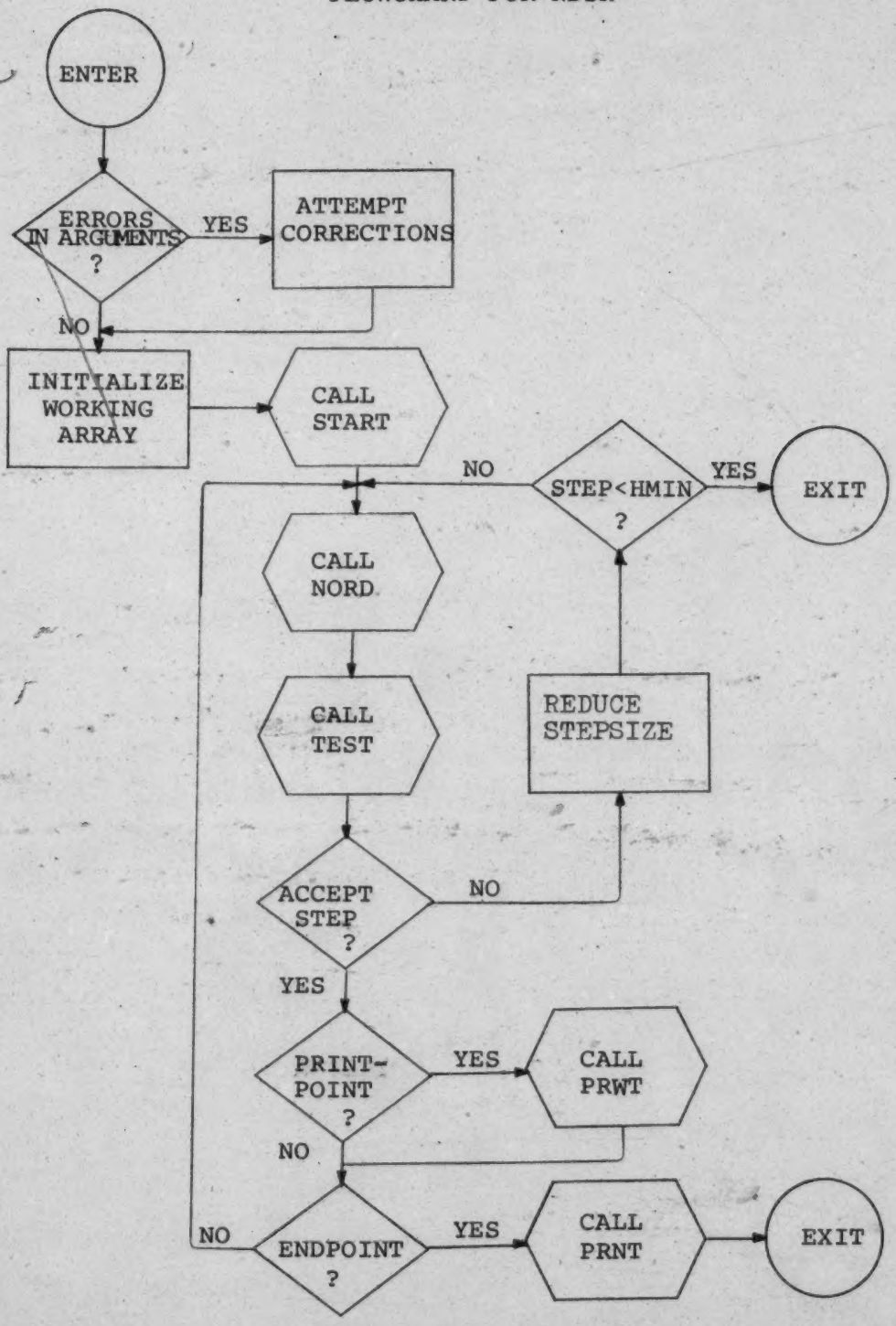


FIGURE 4.2.1

Next, NDSK calls SUBROUTINE START which initializes another part of A by calculating starting values required by Euler's method. Also, START initializes two vectors, CP(12) and D(10), to contain the error constant for the q-th order method in CP(q) and to contain the coefficients C_i^* in D. Furthermore, START is called whenever the stepsize tests fail four times on one step. Such repeated failures indicate an excessive error build-up in the higher derivatives. In this case the higher derivatives are discarted and the integration is started over with Euler's method.

After the initialization, NDSK calls SUBROUTINE NORD which saves the values of A in another array ASAVE and advances the integration one step forward with stepsize HN. NORD is also used to interpolate the solution to the print-points and the endpoint.

Soon as control passes from NORD, NDSK calls SUBROUTINE TEST. When entered, TEST calculates a new stepsize HW and a corresponding order on the basis of the algorithm given in section 3.3. If HW < HN the step will be repeated with, perhaps, lower order; otherwise, the step is accepted.

The sequence CALL NORD - CALL TEST is repeated until one of the following conditions is satisfied:

(i) The stepsize HN is reduced to a value less than HMIN.

(ii) A print-point is reached.

(iii) The endpoint is reached.

At this point NDSK calls SUBROUTINE PRNT to initiate printing the solution value. In case (i) PRNT prints a message. In cases (ii) and (iii) PRNT calls NORD, if necessary, to interpolate the solution to the print-point. When control returns with condition (ii) NDSK returns to execute the sequence CALL NORD - CALL TEST, otherwise, NDSK returns control to the user's calling program.

Finally, we collected all parameters in SUBROUTINE COEFTS which is called by NORD whenever the order changes to place the corresponding parameters in the arrays C and D.

4.3 The Arguments of NDSK

To integrate the system (4.1.1) of differential equations the user must

```
CALL NDSK(TO,TF,TPO,HMAX,HMIN,YO,NEQ,ECT,IODR,KDEL,CQ),
```

where the parameters have the following significance:

TO is a real variable equal to the initial value of the independent variable.

TF is a real variable equal to the final value of the independent variable. TF < TO is permissible.

TPO is a real variable equal to the spacing between print-points. The solution will be interpolated and printed at the points T_k= TO + K*TPO, K=0,1,...,L where L depends on the value of TF.

HMAX is a real variable used as an upper bound for the stepsize.

HMIN is a real variable used as a lower bound for the stepsize. HMIN should be small. In most cases its value should not exceed 10^{-10}, for, the program starts the integration with Euler's method which must use a very small stepsize to produce an accurate solution.

YO is a vector of real variables containing the initial values of the dependent variable. YO should be dimensioned to 10.

NEQ is an integer variable specifying the number of equations in the system (4.1.1).

ECT is a real variable used as error tolerance. ECT appears also in the parameter list of the driving program FEVAL, hence, its value can be changed at each step.

IODR is an integer variable specifying the maximum order to be used.

KDEL is an integer variable specifying the minimum number of steps that the program is to execute between order changes.

CQ is a vector of three real variables related to the bias factors $\beta_{q-1}, \beta_q, \beta_{q+1}$ in section 3.3. That is, we use $CQ(I) = \beta_I$, $I=q-1,q,q+1$. These values may be used to bias the program towards using lower order by assigning a larger value to CQ(3) than to CQ(1) and CQ(2). A reversal of the relation between these factors introduces bias towards higher order. Normally, values in the interval [1.0, 1.5] are used. In most cases CQ = 1.1, 1.2, 1.3 works well.

The variables TO, TF, TPO, HMAX, HMIN, and YO, should be DOUBLE PRECISION with YO dimenioned to 10.

4.4 The Driving Function FEVAL

The user must provide a driving program conforming to

```
SUBROUTINE FEVAL (T,Y,F,TO,TF,HN,ECT)
```

where the parameters have the following meaning:

T is a real variable containing the current value of the independent variable.

Y is a vector of real variables containing the the current values of the components of the dependent variable.

F is a vector of real variables into which the value of f must be placed.

TO,TF are as in NDSK. The ability to change these values is useful to restart the integration at singular points.

HN is a real variable containing the value of the stepsize used to make the current step. Specifying HN overrides the programs choice of the stepsize.

ECT is as in NDSK. For many problems it is useful to specify an error tolerance in terms of Y or F. Several examples of such practice are given in Chapter V.

The variables T,Y,F,TO,T,F, and HN, must be DOUBLE PRECISION with Y and F dimensioned to 10.

4.5 The Tests

The stepsize and the order are determined by

SUBROUTINE TEST .

To calculate the stepsize estimates, TEST expects the highest order scaled derivative of the I-th component of the system (4.1.1) in A(IOR). Test uses the variable IOR for q which is used in the thesis. F_{n+1} and ∇F_{n+1} are expected to be in A(I,14) and A(I,16) respectively. Moreover, the square of the inverse of (τ_n/C_q), used in equation (3.3.5), and the factor q! must be in CP(IOR) and C(2,IOR-1) respectively. Then, TEST calculates the factors

$$T1=CQ(1)*(CP(IOR-1)*C^2(3,IOR-1)*\sum_{I=1}^{NEQ}\left(\frac{A(I,IOR)}{WTS(I)}\right)^2)^{\frac{1}{2*IOR}},$$

$$T2=CQ(2)*(CP(IOR)*C^2(3,IOR-1)*\sum_{I=1}^{NEQ}\left(\frac{A(I,14)}{WTS(I)}\right)^2)^{\frac{1}{2*(IOR+1)}},$$

$$T3=CQ(3)*(CP(IOR-1)*C^2(3,IOR-1)*\sum_{I=1}^{NEQ}\left(\frac{A(I,16)}{WTS(I)}\right)^2)^{\frac{1}{2*(IOR+2)}},$$

where WTS(I) is the maximum value of the I-th component of the approximate solution for $t_0 \le t \le t_n$. TEST uses these factors to calculate the new stepsize

$$HN = HLD/RHO \quad ,$$

where RHO = MIN(T1,T2,T3) and HLD is the stepsize which has been used. The order is then adjusted using the following rule:

(i) If RHO = T1 and IOR is greater than one the order is decreased to IOR-1.

(ii) If RHO = T2 then the order remains IOR.

(iii) If RHO = T3 and IOR is less than IODR the order is increased to IOR+1, where IODR must be given in the calling sequence of NDSK.

4.6 The History of the Program

The first version of our program used the method as given by Nordsieck [16]. This version restricted stepsize changes to the form $h^* = 2^k h$ where k is an integer in the range -5 to 5. It used fixed order methods in the range 1 to 7 and it used the starting procedure outlined by Nordsieck. This program used a very conservative, often extremely small, stepsize.

We then implemented facilities to allow stepsize changes of the form $h^* = \rho h$ where $0.1 \leq \rho \leq 10.0$. At the same time we changed the starting procedure. The new procedure started the integration with Euler's method and increased the order on subsequent steps. These two changes improved the efficiency of the program considerably.

Next, we implemented facilities to let the program change the order. Initially, we had the order-changing mechanism biased towards using low order and used the

vector $\underline{\ell}$ as given by Nordsieck [16]. This version experienced two difficulties. Firstly, the local truncation error estimates were poor and, secondly, the order was changed very frequently. Actually, these two difficulties are related.

The final version of our program allows the order to vary between 2 and 12. It uses the parameters $\underline{\ell}$ as given in section 3.3 and uses the algorithm given in the same section to change the stepsize and order. This algorithm biases the program mildly against order changes. The algorithm of section 3.4 is also implemented.

CHAPTER V

RESULTS OF THE TEST PROBLEMS

5.1 Introduction and Summary

The problems are intended to cover a realistically wide spectrum of problem type. We included some fairly difficult ones, some with singularities in the solution or in its derivatives, some with oscillating solutions, and a class of stiff systems. Most of these problems involve functions which are relatively easy to evaluate and whose analytic solution is available.

When using our program to solve a given differential equation an error tolerance must be specified. In this context it should be noted that we restrict the absolute error rather than a relative error. This choice avoids difficulties when the solution is near zero. Moreover, we restrict the Euclidian norm of a vector of local residuals. Alternatively, facilities can be implemented to assign error tolerances to the individual components of systems of equations. Furthermore, provisions are made to allow the user to specify the error tolerance in terms of the solution of a given equation or in terms of its derivatives.

To specify a particular error tolerance for a given equation requires experience. However, some

guidelines can be given. In general, the error tolerance should be small whenever the error grows strongly and it should be relaxed when the error grows weakly. More specific, if a solution accurate to α significant figures is required then an error tolerance proportional to $10^{-(\alpha+1)}$ will, in most cases, produce a good result. Furthermore, the error tolerance must not be smaller than the uncertainty in the driving function. That is, if the value of the driving function is given accurately to α decimal places then the error tolerance must be larger than $10^{-\alpha}$.

For particular examples of choices of error tolerances we refer the reader to the examples.

Comparing results obtained with different programs is, in general, not easy. In particular, such comparison requires a basis which is difficult to define. However, we believe that the number of function evaluations required to produce a solution of specified accuracy is a valid criteria. On this basis we compared our results with those obtained by Gabel [4]. We found that our results were obtained slightly more efficiently for equations having solutions which decrease in absolute value, but the reverse was true for equations having increasing solutions. Moreover, we observed that our program increased the order and stepsize slower which results in a slight overall loss of efficiency.

5.2 Equations of Exponential Functions

The two simple initial value problems

$$y^{(1)} = y \ , \quad y(0) = 1.0 \ , \tag{5.2.1}$$

$$y^{(1)} = -y \ , \quad y(0) = 1.0 \ , \tag{5.2.2}$$

which have the analytic solutions $y = e^{t}$ and $y = e^{-t}$ appear quite often in reports on differential equation solvers.

We obtained solutions at t = 20 using, for equation (5.2.1), the error tolerance

$$ECT = 10^{-K} \ , \tag{5.2.3}$$

and, for equation (5.2.2),

$$ECT = 10^{-K}|Y_n| \ , \quad K=1,2,3,\ldots,10 \ . \tag{5.2.4}$$

The results are shown in Tables 5.2.1 and 5.2.2.

Equation (5.2.2) is an example of the usefulness of variable error tolerances. To see this, let us assume that (5.2.2) is solved using an absolute error tolerance $ECT = 10^{-5}$. Such bound on the absolute error together with the initial condition y(0) = 1.0 should produce, near t = 0, a numerical solution with about 5 significant figures accuracy. However, $y(20) \approx 10^{-8}$, hence, using the same absolute bound near t = 20 cannot

Results of $y^{(1)} = y$, $y(0) = 1.0$, at $t = 20$

K	RELATIVE ERROR	FUNCTION EVALUATIONS	STEPS
1	-0.1933E 01	68	24
2	-0.1038E 00	88	31
3	-0.1295E-01	144	47
4	-0.1516E-02	179	58
5	-0.1814E-03	202	70
6	-0.3122E-04	306	105
7	-0.9561E-06	320	107
8	-0.2701E-06	400	134
9	-0.1152E-07	486	165
10	-0.2233E-08	658	219

TABLE 5.2.1

Results of $y^{(1)} = -y$, $y(0) = 1.0$, at $t = 20$

K	RELATIVE ERROR	FUNCTION EVALUATIONS	STEPS
1	0.1160	109	38
2	-0.1077	147	49
3	0.1842E-02	158	71
4	-0.7031E-03	237	84
5	0.2785E-04	320	118
6	0.4948E-05	460	167
7	-0.1779E-06	481	178
8	-0.8393E-10	589	231
9	0.7715E-08	579	255
10	0.8208E-09	771	318

TABLE 5.2.2

be expected to produce a solution at t = 20 having any accuracy at all. Therefore, we use the error tolerance (5.2.4) which is proportional to the magnitude of the solution.

5.3 Sine-Cosine Differential Equations

The initial value problem defined by the system of differential equations

$$\underline{y}^{(1)} = A\,\underline{y} \quad , \tag{5.3.1}$$

where

$$A = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \quad ,$$

and

$$\underline{y}(0) = \text{col}(0.0,\ 1.0),$$

is also commonly used to test integration routines.

It can be shown that the solution of the magnified error equation of equation (5.3.1) using Euler's method oscillates. Hence, by integrating equation (5.3.1) over a long interval one can check the integration routines ability to stay with an oscillating solution.

We obtained a solution for equation (5.3.1) at t = 200 using the absolute error tolerance ECT = 10^{-7},

an upper bound HMAX = 5.0, and a lower bound HMIN = 10^{-13}, for the stepsize.

Table 5.3.1 shows the absolute errors $\varepsilon_{n,i} = y_{n,i} - Y_{n,i}$, i=0,1 and the number of steps taken from t = 0. For simplicity we show $\varepsilon_{n,i} \times 10^6$.

Results of the Sine-Cosine Problem

t	$\varepsilon_{n,0} \times 10^6$	$\varepsilon_{n,1} \times 10^6$	STEPS
10	0.1783	0.2780	81
50	-0.6901	-0.1266	282
100	-0.9639	0.2582	534
200	0.9283	-0.5974	1037

TABLE 5.3.1

The program evaluated f about twice for each step taken, namely, 2106 times over the interval $0 \leq t \leq 200$. The relatively large number of steps taken to reach the point t = 10 indicates that the program increases the stepsize and order slowly.

5.4 A Non-Linear Problem

The initial value problem defined by

$$\underline{y}^{(1)} = A\,\underline{y} \quad , \tag{5.4.1}$$

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -r^{-3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -r^{-3} & 0 \end{pmatrix} ,$$

$$r = (y_1^2 + y_3^2)^{\frac{1}{2}} ,$$

$$\underline{y}(0) = \text{col}(1.0,\ 0.0,\ 0.0,\ 1,0) ,$$

is a non-linear system which has the analytic solution

$$\underline{y}(t) = \text{col}(\cos t,\ -\sin t,\ \sin t,\ \cos t) .$$

We obtained a solution for the system (5.4.1) at $t = 100$ using the absolute error tolerance $ECT = 10^{-7}$. $HMAX = 2.0$ and $HMIN = 10^{-6}$ were used to bound the stepsize.

Table 5.4.1 contains the error $\varepsilon_{n,i}$, $i=0,1,2,3$ at $t = 10, 50, 100$. We also include the values $R_n = (Y_{n,1}^2 + Y_{n,3}^2)^{\frac{1}{2}}$. The error growth in the solution Y_n is partly due to using R_n in place of r which should be equal to 1.0.

Results of the Non-Linear Problem

t	$\varepsilon_{n,0}$	$\varepsilon_{n,1}$	$\varepsilon_{n,2}$	$\varepsilon_{n,3}$	R_n	STEPS
10	0.2467E-05	-0.4087E-05	0.3522E-05	-0.2590E-05	0.9999998	105
50	0.2294E-04	-0.9583E-04	0.9659E-04	0.2748E-04	1.000003	360
100	0.2760E-03	-0.4798E-03	0.4854E-03	0.2862E-03	1.000008	679

TABLE 5.4.1

5.5 The Satellite Orbit Problem

The system of equations for this problem is

$$y_0^{(1)} = y_1 \quad ,$$

$$y_1^{(2)} = y_0 y_3^2 - 0.13993741 \times 10^{17}/y_0^2 \quad , \qquad (5.5.1)$$

$$y_2^1 = y_3 \quad ,$$

$$y_3^{(1)} = -2y_1 y_3/y_0 \quad ,$$

with the initial conditions

$$y_0(0) = 0.21155 \times 10^8 \quad ,$$

$$y_1(0) = 0.0 \quad , \qquad (5.5.2)$$

$$y_2(0) = 0.0 \quad ,$$

$$y_3(0) = 0.1625 \times 10^{-2} \quad .$$

The components 0 to 3 of the system (5.5.1) represent respectively the radius, the radial velocity, the azimuthal angle, and the azimuthal velocity of an orbit. The units are feet, feet per second, radians, and radians per second.

The system (5.5.1) defines a highly non-linear system of equations which has a periodic solution with period T = 52415.256 seconds; hence, the solution vector, except the third component, should return to the initial

condition (5.5.2). The third component should attain the value $2K\pi$ after K periods.

We solved problem (5.5.1) over 10 orbits using the absolute error tolerance ECT = 10^{-12}. We specified HMAX = 2000 and HMIN = 10^{-16} and found that the program varied the stepsize from a low value of 0.145 to a high value of 2000.

Table 5.5.1 shows, at the completion of each orbit, the differences di, i=0,1,2,3, between the initial condition (5.5.2) and the numerical solution $\underline{Y}$, except for the third component from which $2K\pi$ is subtracted. In addition, we report the number of function evaluations NOF.

Results of the Satellite Orbit Problem

NUMBER OF ORBITS	d_1	d_2	$d_3 \times 10^4$	$d_4 \times 10^8$	NOF
1	-0.807E 01	-0.2296	-0.1499	0.1023	1228
2	-0.2655E 02	-0.7168	-0.4693	0.3161	2494
3	-0.4493E 02	-0.8785	-0.0530	0.5314	3769
4	-0.5599E 02	0.1180E 01	0.7876	0.6639	5182
5	-0.7318E 02	0.1729E 01	1.1542	0.8719	6415
6	-0.1034E 03	0.2975E 01	1.9784	1.2304	7624
7	-0.1083E 03	0.4618E 01	3.0657	1.2900	8887
8	-0.1294E 03	0.6402E 01	4.2478	1.5429	10354
9	-0.1609E 03	0.8784E 01	5.8232	1.9171	11449
10	-0.1639E 03	0.1133E 02	7.5058	1.9592	12772

TABLE 5.5.1

5.6 A Stiff Linear System

To test the programs performance on a stiff system we integrated

$$\underline{y}^{(1)} = A\,\underline{y}\ ,\quad \underline{y}(0) = \mathrm{col}(2.0,\ 0.0)\ , \qquad (5.6.1)$$

where

$$A = -\frac{1}{2}\begin{pmatrix} 1+c & 1-c \\ 1-c & 1+c \end{pmatrix}\ ,$$

which has the analytic solution

$$y_0(t) = e^{-t} + e^{-ct}\ ,$$

$$y_1(t) = e^{-t} - e^{-ct}\ .$$

We solved equation (5.6.1) over the interval $0 \le t \le 1$ with c=4, 16, 64, 256. We used the absolute error tolerance ECT = 2^{-25} and specified HMAX = 2^{-3} and HMIN = 2^{-11}.

The results in Table 5.6.1 show the value used for c, the relative error in each component of the numerical solution, and the number of function evaluations NOF.

Results of a Stiff System

c	RELATIVE ERROR		NOF
	in Y_0	in Y_1	
4	0.2947E-07	-0.6719E-08	138
16	-0.1565E-07	0.1502E-07	215
64	-0.1042E-06	-0.8599E-07	368
256	0.1524E-05	0.1522E-05	864

TABLE 5.6.1

5.7 The Solution of $y^{(1)} = 20y/t$

Nordsieck [16] discusses the problem

$$y^{(1)} = 20y/t \ , \quad y(\tfrac{1}{2}) = 2^{-21} \ , \qquad (5.7.1)$$

which has the analytic solution $y(t) = t^{20}/2$. Lewis and Stovall [15] also report a solution.

We integrated the equation (5.7.1) for $\frac{1}{2} \le t \le 1$ using the absolute error constants ECT = 2^{-K}, K=24, 32, 40, 48, and specified HMAX = 2^{-4}.

The results are reported in Table 5.7.1 where we list ECT, Y(1), the error relative to the true solution $y(1) = \frac{1}{2}$, the number of steps, and the number of function evaluations NOF.

Results of $y^{(1)} = 20y/t$ at $t = 1.0$

ECT	Y(1)	RELATIVE ERROR	STEPS	NOF
2^{-24}	0.52036475	-0.4073E-01	48	384
2^{-32}	0.50042355	-0.8471E-03	76	447
2^{-40}	0.50000413	-0.8262E-05	126	669
2^{-48}	0.50000002	-0.4743E-07	289	1598

TABLE 5.7.1

The equation (5.7.1) has a strongly increasing solution; hence, care must be taken to keep the early errors low. We did not calculate solutions using sufficiently many values for ECT to graph the error versus the number of function evaluations to support the conclusion that the programs performance increases with decreasing error bounds. However, the results of Table 5.7.1 are consistent with our general observation that such is the case.

5.8 A Problem with Discontinuous f

The following two examples are designed to show how differential equations having discontinuities in f are handled. The equations are:

$$y^{(1)} = \begin{cases} 3t^2 & , \text{ if } t \leq 1 \ , \\ -ty^2 & , \text{ if } t > 1 \ , \end{cases} \qquad (5.8.1)$$

$y(0) = 1$, and

$$y^{(1)} = 20(4-t^2)\text{sign}(t)/(4+t^2) \ , \qquad (5.8.2)$$

$y(-4) = 4$.

Equation (5.8.1) has a singularity at $t = 1$ where the value of f changes by a factor of 1. The singularity in (5.8.2) occurs at $t = 0$ with a jump of 10 in f.

We integrated both equations across these singular points using error tolerances

$$ECT = 10^{-9} \ ,$$

and

$$ECT = 10^{-8}|f_n| + 10^{-11} \ ,$$

respectively for equations (5.8.1) and (5.8.2). The results are shown in Table 5.8.1.

Results of Two Equations with Discontinuous f

EQUATION	t	Y	y	FUNCTION EVALUATIONS
(5.8.1)	3.0	0.2222222295	0.22222222	394
(5.8.2)	4.0	4.0000004	4.0	471

TABLE 5.8.1

We observed that in both cases the program reduced the order to that of Euler's method and the stepsize to $O(10^{-10})$ before passing the singular point.

5.9 The Solution of $y^{(1)} = -2ty^2$

The initial value problem

$$y^{(1)} = -2ty^2 \quad , \quad y(0) = 1.0 \quad , \tag{5.9.1}$$

is a simple non-linear differential equation whose analytic solution $y(t) = (1+t^2)^{-1}$ is difficult to interpolate with a polynomial.

We solved problem (5.9.1) for $0 \leq t \leq 10$ using the error tolerances

$$ECT = 0.2 \times 10^{-K}|f_n| + 10^{-16} \quad , \tag{5.9.2}$$

and

$$ECT = 0.2 \times 10^{-K}|y_n| + 10^{-16} \quad , \tag{5.9.3}$$

K=3,5,7. We specified HMAX = 1.0 and HMIN = 10^{-21}.

The results, listed in Table 5.9.1, show the error relative to the true solution and the number of function evaluations NOF.

Results of $y^{(1)} = -2ty^2$, $y(0) = 1.0$, at $t = 10$

ECT	K = 3		K = 5		K = 7	
	RELATIVE ERROR	NOF	RELATIVE ERROR	NOR	RELATIVE ERROR	NOF
(5.9.2)	0.6665E-04	133	-0.1088E-05	232	0.1127E-08	407
(5.9.3)	-0.1164E-04	154	0.2090E-06	304	-0.1673E-08	375

TABLE 5.9.1

5.10 Solve for y in $y(t) = \int_0^{\pi} \sin(t \sin z)dz$

The integral

$$y(t) = \int_0^{\pi} \sin(t \sin z)dz \qquad (5.10.1)$$

can be evaluated by solving the system of differential equations

$$y_0^{(1)} = y_1 \quad ,$$
$$y_1^{(1)} = (2 - y_1)/t - y_o \quad , \qquad (5.10.2)$$

$y_0(0) = 0$, $y_1(0) = 0$, and $y_1^{(1)}(0) = 0$.

Inspecting the system (5.10.2) shows that the second component equation causes difficulties when t is near zero. If the error tolerance is small, then, near t = 0, the program will choose an extremely small step-size. Therefore, we used the error tolerance ECT = 0.1 for the first step and continued with

$$ECT = 0.2 \times 10^{-5} \times ((Y_o^{(1)})^2 + (Y_1^{(1)})^2) + 10^{-8}/t \ .$$

The results are reported in Table 5.10.1. We noted that a relatively large number of steps were taken to reach the point t = 10 which indicates that the program had difficulties starting the integration.

Results of the System (5.10.2)

t	Y(t)	FUNCTION EVALUATIONS	STEPS
10	0.37327	302	92
50	-0.26813	674	216
100	-0.22268	1142	371

TABLE 5.10.1

5.11 Problems with Singularities

The problems in this section, taken from Gabel [4], have singularities in the first or higher derivatives. The equations are

$$y^{(1)} = (t-1)^{4/3} \ , \quad y(0) = \frac{3}{7} \ , \qquad (5.11.1)$$

$$y^{(1)} = (t-1)^{1/3} \ , \quad y(0) = \frac{3}{4} \ , \qquad (5.11.2)$$

$$y^{(1)} = (t-1)^{-1/3} \ , \quad y(0) = \frac{3}{2} \ , \qquad (5.11.3)$$

$$y^{(1)} = \begin{cases} 0 \ , & \text{when } t < 0 \ , \\ e^t - 1 - t - \frac{t^2}{2} \ , & \text{when } t \geq 0 \ . \end{cases} \qquad (5.11.4)$$

We note that the derivatives of equation (5.11.1), starting with the third, have a pole at t = 1. Therefore, the order of the integration formulae must be lowered in order to pass the point t = 1. Equation (5.11.2) has similar difficulties. Equation (5.11.3) has a pole in f and in all its derivatives at t = 1. Equation (5.11.4) has derivatives with finite jumps at t = 0.

We obtained solutions for equations (5.11.1) to (5.11.3) over the interval $0 \leq t \leq 2$ and for (5.11.4) over the interval $-\frac{1}{2} \leq t \leq 1$. We used the following error tolerances for equations (5.11.1) to (5.11.4) respectively:

$$ECT = 10^{-10}|f_n| + 10^{-12} \ , \qquad (5.11.5)$$

$$ECT = 10^{-10}|f_n| + 10^{-12}\, t/f_n^2 \ , \qquad (5.11.6)$$

$$ECT = 10^{-9}|f_n| + 10^{-11}\, f_n^4 \ , \qquad (5.11.7)$$

$$ECT = 10^{-9} \ . \qquad (5.11.8)$$

The results shown in Table 5.11.1 include the numerical solution at the endpoint, the error relative to the true solution, and the number of function evaluations NOF.

Results of Equations (5.11.1) to (5.11.4)

EQUATION	$Y(t_f)$	RELATIVE ERROR	NOF
(5.11.1)	0.42857128	0.3374E-06	353
(5.11.2)	0.74999989	0.1389E-06	965
(5.11.3)	1.50021613	-0.1441E-03	448
(5.11.4)	1.05161517	0.8371E-08	109

TABLE 5.11.1

5.12 The Hull and Cremer Problems

Hull and Cremer [10] list 17 equations which they used to test the efficiency of predictor-corrector schemes. These equations are listed in Table 5.12.1.

We solved these equations and report the results in Table 5.12.2. The initial conditions at t = 0 were obtained from the analytic solution. For each problem we list the error tolerance ECT that had been used, the upperbound HMAX for the stepsize, the error relative to the true solution at t = 40, and the number of function evaluations NOF.

The Hull and Cremer Problems

PROBLEM	$y^{(1)}(t)$	$y(t)$
A	$-y + 10 \sin 3t$	$\sin 3t + 3 \cos 3t$
B	$-y + 2 \sin t$	$\sin t - \cos t$
C	$y + 2 \sin t$	$-\sin t - \cos t$
D	$-3y + 10 \sin t$	$3 \sin t - \cos t$
E	$y \cos t$	$e^{\sin t}$
F	$y \cos^2 t$	$e^{(t/2+\sin 2t/4)}$
G	$(y-\sin t)\log(1+t/40)+\cos t$	$\sin t$
H	$y(y - \sin t) + \cos t$	$\sin t$
I	$y(y - \sin^2 t) + \sin 2t$	$\sin^2 t$
J	$-ty/(4t + 16)$	$(t+4)e^{-t/4}$
K	$-y^3$	$(2t+2)^{-\frac{1}{2}}$
L	$y/4$	$e^{t/4}$
M	$y - 2t/y$	$(2t + 1)^{-\frac{1}{2}}$
N	$y/40$	$e^{t/40}$
O	y^2	$1/(40.01 - t)$
P	$y^{\frac{1}{2}}$	$(5 + t/2)^2$
Q	$(1 + t^2)/2(2500 - t^2)^{\frac{1}{2}}$	$((50 + t)/(50 - t))^{\frac{1}{2}}$

TABLE 5.12.1

Results of the Hull and Cremer Problems

PROBLEM	ECT	HMAX	RELATIVE ERROR	NOF
A	0.3162E-05×|Y|	0.25	0.1950E-04	1916
B	0.1414E-04×|Y|	0.25	0.5958E-06	484
C*	0.1414E-04×|Y|	0.25	-0.8912E 14	551
D	0.3612E-04×|Y|	0.08	-0.7510E-06	754
E	0.2718E-04×|Y|	0.25	0.7304E-04	713
F	0.1E-04×|Y|	0.25	-0.4828E-01	584
G*	0.1E-04×|Y|	0.83	-0.1535E 03	507
H	0.1E-04×|Y|	0.83	-0.1832E-03	724
I*	0.1E-04×|Y|	0.25	0.1128E 01	925
J	0.4E-05×|Y|	1.00	-0.2281E-04	145
K	0.1414E-04×|Y|	0.17	0.1084E-05	280
L	0.1E-04×|Y|	1.00	-0.2137E-03	140
M*	0.1E-04×|Y|	0.13	-0.1130E 15	689
N	0.1E-04×|Y|	10.00	-0.4624E-05	53
O	0.1E-07×|Y|	0.01	-0.1019E-04	4155
P	0.1E-04×|Y|	2.50	0.1034E-07	53
Q	0.1E-05×|Y|	2.50	-0.5024E-05	82

TABLE 5.12.2

Four of the equations listed above, these are starred, are unstable. Our program integrated all four of these problems to the endpoint.

Our program and Gabel's were run on different machines, hence, comparisons of results are difficult to make. Yet, we have observed consistently that Gabel's program produced better solutions for increasing functions, whereas, ours produced better results for functions decreasing in absolute value.

5.13 The Solution of $y^{(1)} = y \log y/t$

The family of curves $y = e^{ct}$ satisfying the differential equation

$$y^{(1)} = y \log y/t \qquad (5.13.1)$$

has a common point at $t = 0$. An initial condition, given at any point $t \neq 0$, uniquely defines a particular curve. However, due to the common point at $t = 0$ and the exponential form of the family of solution curves errors introduced near $t = 0$ are greatly magnified. Hence, near the critical point $t = 0$ the error control must be very tight.

We solved equation (5.13.1) for $-5 \leq t \leq 5$ using the error tolerance

$$ECT = 10^{-12}(Y_n^2 + Y_n^{-2}) \quad .$$

Note that ECT attains a minimum when $Y_n = 1$, that is, when $t = 0$. The results are given in Table 5.13.1.

Results of $y^{(1)} = y \log y/t$

t	RELATIVE ERROR	STEPS	FUNCTION EVALUATIONS
-4	-0.1104E-08	44	116
-3	-0.2311E-08	62	170
-2	-0.4183E-08	80	224
-1	-0.9361E-08	97	286
0	-0.9128E 11	111	329
1	-0.7060E-06	121	359
2	-0.1626E-05	126	372
3	-0.3087E-05	129	381
4	-0.7633E-05	132	390
5	-0.2583E-04	134	399

TABLE 5.13.1

5.14 The Solution of a Bessel Equation

To test a differential equation solver over a long interval Nordsieck [16] used the equation

$$t^2 y^{(2)} + t y^{(1)} - (256 - t^2) y = 0 \quad , \qquad (5.14.1)$$

where $y(t) = J_{16}(t)$ which is a Bessel function of order

16. $J_{16}(t)$ is small in absolute value at t = 6. However, its value increases rapidly when passing t = 6 and oscillates more than 1000 times over the interval $6 \le t \le 6136$.

We integrated (5.14.1) in the form

$$y_0^{(1)} = y_1 \ , \qquad y_0(6) = 0.1201950 \times 10^{-5} \ ,$$

$$y_1^{(1)} = (\frac{256}{t^2} - 1)y_0 - y_1/t \ , \quad y_1(6) = 0.2986480 \times 10^{-5} \ ,$$

using the error tolerance

$$ECT = 10^{-7}(||\underline{y}_n|| + ||\underline{y}_n^{(1)}||) \ .$$

We specified HMAX = 5 and HMIN = 10^{-11}. The results are shown in Table 5.14.1.

Results of a Bessel Equation

t	y	Y	ERROR	STEPS	FUNCTION EVAL'S
6136	-0.0097446808	-0.009745831	-0.000001150	23869	69464

TABLE 5.14.1

The results compare well with Gabel's [4]; but are not as good as Nordsieck's [16]. Actually, Nordsieck obtained his solution using stepsizes of the form 2^K where

K is integral. Such choice of stepsizes reduces the round-off error considerably.

5.15 The Solution of $y^{(1)} = a^2b/(t^2+a^2)$

Selecting HMAX for problems of the form

$$y^{(1)} = a^2b/(t^2+ a^2) \qquad (5.15.1)$$

is quite important, particularly, if a is small. Equation (5.15.1) has a general solution of the form

$$y(t) = a \arctan (t/b) + a \arctan(-t_0/b)$$

which has a graph as illustrated in Figure 5.15.1.

Graph of the Equation (5.15.1)

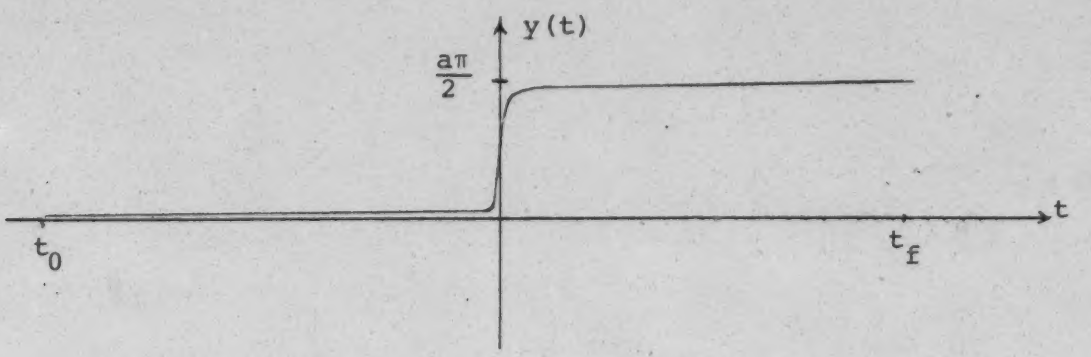


FIGURE 5.15.1

This example involves searching the t axis for an extremely narrow interval in which f peaks. If HMAX is larger than this interval then the program will likely overstep it without noticing the sudden change and produce a useless result.

We solved this problem with $a = 2^{-30}$ and $b = 2^{7}$ for $-\frac{1}{2} \leq t \leq \frac{1}{2}$. We used $y(-\frac{1}{2}) = 0$, error tolerance $ECT = 10^{-13}$, and $HMAX = 2^{-K}$, where K=8,9,10,11.

The results for each value of K together with the solution $Y(\frac{1}{2})$, the error at $t = \frac{1}{2}$, the number of function evaluations NOF, and the number of steps, are given in Table 5.15.1.

The analytic solution is $y(\frac{1}{2}) = 0.37450728 \times 10^{-6}$.

Results of Equation (5.15.1)

K	$Y(\frac{1}{2})$	ERROR	NOF	STEPS
8	0.37912902E-12	0.3745E-06	276	258
9	0.90735639E-12	0.3745E-06	527	513
10	0.11877811E-11	0.3775E-06	1041	1025
11	0.37451038E-06	-0.3351E-11	2752	2295

TABLE 5.15.1

5.16 The Solution of ODE's with Error Estimates

Gabel [4] gives an error analysis for the initial value problem defined by the simple system of differential equations

$$y_i^{(1)} = \begin{cases} 1 & , \text{ if } i=1 \\ iy_{i-1} & , \text{ if } i=2,3,4,\ldots,K \ , \end{cases} \tag{5.16.1}$$

$y_i(0) = 0$. The system (5.16.1) has the general solution $y(t) = t^i$. The analysis concludes that if an absolute error tolerance τ is used then the error $\varepsilon_{n,i}$ in the numerical solution for the i-th component of (5.16.1) should obey

$$|\varepsilon_{n,i}| \leq 2\tau t^{i+1}(2^i - 1)/(i+1) \ . \tag{5.16.2}$$

Moreover, if a relative error tolerance $\tau_{n,i} = \tau|Y_{n,i}|$ is used then

$$|\varepsilon_{n,i}| \leq i\tau t^{i+1}/(i+1) \tag{5.16.3}$$

ought to hold.

We integrated (5.16.1) using K=8, $\varepsilon=10^{-8}$, and allowed the order to vary up to and inclusive the value 9. Then, we repeated the calculations with the order restricted to values less than or equal to 6 which is less than the degree of the solution polynomials of the

last two components of (5.16.1). Tables 5.16.1 and 5.16.2 list for i=2,4,8 the value of the analytic solutions Y_i at t=2 and t=20, the value of the bounds (5.16.2) and (5.16.3) using the final value of t, the absolute error, and the number of function evaluations NOF.

The results in Tables 5.16.1 and 5.16.2 show a much smaller error than bound where i is small. However, when i equals 8 the errors are approaching the values of the bounds. This phenomenon is due to the fact that our program does not bound the errors of the individual components of a system of equations; it bounds the Euclidian norm of the vector of residuals. Consequently, some variation of the error over the components of a system of equations is possible.

Results of Equation (5.16.1), K = 8, ORDER $\leq$ 9, at t = 2

i	Y(2)	RELATIVE TOLERANCE		ABSOLUTE TOLERANCE	
		ERROR	BOUND	ERROR	BOUND
2	0.400E 01	0.1554E-14	0.5333E-07	0.1110E-14	0.1600E-06
4	0.800E 01	0.1493E-07	0.2560E-06	0.1512E-07	0.1920E-05
8	0.256E 03	0.4982E-04	0.4551E-05	0.9980E-01	0.2901E-03
NOF		207		204	

TABLE 5.16.1

Results of Equation (5.16.1), K = 8, ORDER ≤ 6, at t = 20

i	y(20)	RELATIVE TOLERANCE		ABSOLUTE TOLERANCE	
		ERROR	BOUND	ERROR	BOUND
2	0.200E 02	0.5116E-12	0.5333E-04	0.3411E-12	0.1600E-03
4	0.800E 02	0.1197E-06	0.2560E-01	0.1216E-06	0.1920E 00
8	0.256E 03	0.1158E 06	0.4551E 04	0.9444E 04	0.2901E 06
NOF		512		763	

TABLE 5.16.2

5.17 The Parabolic Surface

An example, given by Gabel [4], to test a differential equation solver's ability to handle discontinuities is the system

$$y_0^{(1)} = \begin{cases} 0 , & \text{if } y_0^2 + y_1^2 > t , \\ y_0 + ty_1 , & \text{otherwise} , \end{cases}$$

$$y_1^{(1)} = \begin{cases} 0 , & \text{if } y_0^2 + y_1^2 > t , \\ y_1 - ty_o , & \text{otherwise} , \end{cases} \qquad (5.17.1)$$

Note that the parabloid $y_0^2 + y_1^2 = t$ separates the Euclidean 3-space into two regions E_1 and E_2 as indicated in Figure 5.17.1 and that any solution of the system

(5.17.1) started in E_1, except $y_0 = y_1 = 0$, takes a finite jump when meeting the surface $y_0^2 + y_1^2 = t$ which separates E_1 from E_2 .

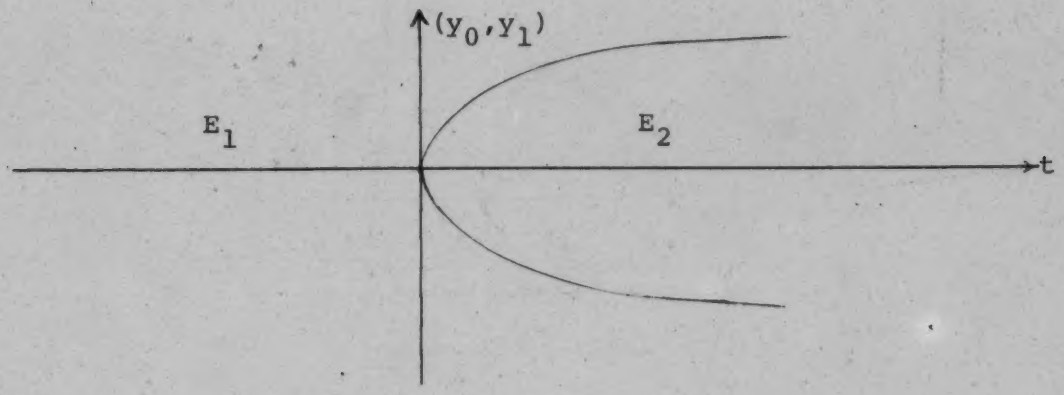


FIGURE 5.17.1

We attempted to calculate several solutions of (5.17.1) using various initial values at t = -1. Our routine had no difficulty extending solutions across the mentioned surface whenever $Y_0^2 + Y_1^2 < \frac{1}{2}$. However, all solutions were quickly terminated whenever $Y_0^2 + Y_1^2 \geq \frac{1}{2}$.

Some of the results obtained are as follows:

(i) Using the initial condition $y_0(-1) = y_1(-1) = 0$ leads to the trivial case $y_0(t) = y_1(t) = 0$. The calculated solutions followed the line $Y_0 = Y_1 = 0$.

(ii) Using $y_0(-1) = y_1(-1) = 0.25$ the solution curve entered the region E_2 without difficulty; but, the curve terminated when it touched the surface from inside E_2 at the point $(t,Y_0,Y_1) = (2.41729, -0.83270, -1.31297)$. At this point f had been evaluated 239 times.

(iii) Using $y_0(-1) = y_1(-1) = 0.5$ the solution curve encountered the surface at $(t,Y_0,Y_1) = (0.87823, 0.32522, 0.87889)$ and terminated. The function f had been evaluated 70 times.

5.18 The Ricatti-Bessel Equation

The Ricatti equation

$$w^{(1)} = w^2/3 + 3t^2 , \quad w(0) = 0 , \qquad (5.18.1)$$

has the solution

$$w(t) = 3t\, J_{3/4}(t^2/2)/J_{-1/4}(t^2/2) ,$$

where $J_{3/4}(z)$ and $J_{-1/4}(z)$ are Bessel functions of orders 3/4 and -1/4 respectively.

The function w(t) has a pole at any point t_i such that $J_{-1/4}(t_i^2/2) = 0$; hence, a differential equation solver should not integrate past such points t_i. Therefore, it can be used to locate the zeros of the function

$J_{-1/4}(z)$. To locate the zeros of both functions, $J_{3/4}(z)$ and $J_{-1/4}(z)$, we used the algorithm:

(i) Starting at $t = t_i$ integrate the equation (5.18.1) to $t = t_{i+1}$ at which point the program reduces the stepsize to a value less than the specified HMIN.

(ii) Take the value $z_{i+1} = t_{i+1}^2/2$ as an approximation of a zero of $J_{-1/4}(z)$.

(iii) Switch to the related equation

$$u^{(1)} = -\frac{1}{3} - 3u^2t^2 \ , \ u(t_{i+1}) = \frac{1}{w(t_{i+1})} \ , \qquad (5.18.2)$$

which is obtained from (5.18.1) by applying the transformation $u(t) = w^{-1}(t)$.

(iv) Starting at $t = t_{i+1}$ integrate equation (5.18.2) to the point $t = t_{i+2}$ where the stepsize is again reduced to a value less than HMIN. Take $z_{i+2} = t_{i+2}^2/2$ to be an approximation of a zero of $J_{3/4}(z)$. Replace t_i by t_{i+2} and return to step (i).

Using the above algorithm with HMIN = 10^{-4} and the error tolerance ECT = 10^{-7} we calculated approximations to the first seven zeros of $J_{3/4}(z)$ and $J_{-1/4}(z)$. Tables 5.18.1 and 5.18.2 list these zeros together with the true zeros γ_i which are obtained from the tables and the number of function evaluations NOF.

The large number of function evaluations is due to the facts that the integration must be restarted whenever a zero is located and that our program has the tendency to reduce the order and stepsize before giving up.

Table 5.18.3 shows the numerical solution w(t) at the points $t = 2\sqrt{K}$, K=1,2,3,4,5. The true values w(t) are taken from Gabel [4].

Roots of $J_{-1/4}(t)$, $0 \leq t \leq 25$

i	t_i	γ_i	NOF
1	2.006294	2.006299	1228
2	5.123050	5.123062	1123
3	8.257929	8.257951	970
4	11.396436	11.396468	1090
5	14.536256	14.536299	1074
6	17.676697	17.676753	1060
7	20.817481	20.817550	1064

TABLE 5.18.1

Roots of $J_{3/4}(t)$, $0 \le t \le 25$

i	t_i	γ_i	NOF
1	3.490998	3.491008	971
2	6.652614	6.652635	1100
3	9.801582	9.801612	970
4	12.946992	12.947034	943
5	16.090914	16.090969	939
6	19.234074	19.234114	920
7	22.376790	22.376871	906

TABLE 5.18.2

Results of the Ricatti Equation

z	$z^2/2$	COMPUTED w(z)	TRUE w(z)	NOF
2	2.0	953.3120	950.*	580
$2\sqrt{2}$	4.0	4.5665	4.5667	1199
$2\sqrt{3}$	6.0	-8.1924	-8.1943	1821
4.0	8.0	45.8587	45.8664	2384
$2\sqrt{5}$	10.0	2.6838	2.6846	3081

TABLE 5.18.3

The value marked * in Table 5.18.3 is in error.

5.19 An Oscillating Solution

The initial value problem

$$y_0^{(1)} = -2ty_0 \log y_1 \quad , \quad y_0(0) = e \quad ,$$

$$y_1^{(1)} = 2ty_1 \log y_0 \quad , \quad y_1(0) = 1 \quad , \qquad (5.19.1)$$

is used by Fehlberg [8] to test Runge-Kutta methods of orders 5 to 8. The system (5.19.1) has the analytic solution

$$y_0(t) = \text{EXP}(\cos t^2) \quad ,$$

$$y_1(t) = \text{EXP}(\sin t^2) \quad ,$$

hence, both components of the solution vector oscillate with period $\sqrt{\pi}$ and amplitude $e - e^{-1}$.

Fehlberg [8] uses a local truncation error tolerance equal to 10^{-16}. In our environment such small error tolerance introduces serious roundoff error problems. Hence, we experimented to determine the smallest values which when used as error tolerances allows our routine to perform reasonably efficient. The two values

$$\text{ECT} = 10^{-15} \qquad (5.19.2)$$

and

$$\text{ECT} = 10^{-14}(Y_0^2 + Y_1^2) \qquad (5.19.3)$$

satisfied such requirement.

We integrated (5.19.1) over the interval $0 \le t \le 5$ and report the results in Table 5.19.1.

The error in our solutions is larger than in Fehlberg's by a factor of about 10. However, we counted 6033 and 7186 function evaluations using respectively the error tolerances (5.19.2) and (5.19.3) which is considerably below the values reported in [8]. Thus, this result supports the conjecture that multistep and related methods are more efficient if f is expansive to evaluate.

Results of an Oscillating Problem

t	ERROR TOLERANCE	RELATIVE ERROR IN		STEPS
		Y_0	Y_1	
1.0	(5.19.2)	-0.1306D-13	0.3676D-13	237
	(5.19.3)	-0.1785D-13	0.5839D-13	404
2.0	(5.19.2)	-0.1001D-12	0.4168D-13	532
	(5.19.3)	-0.1589D-12	0.4407D-13	886
3.0	(5.19.2)	0.1060D-12	0.3923D-12	1043
	(5.19.3)	0.7259D-13	0.4213D-12	1383
4.0	(5.19.2)	-0.3059D-12	0.7149D-12	1617
	(5.19.3)	-0.3971D-12	0.8413D-12	2001
5.0	(5.19.2)	-0.8900D-13	-0.1565D-11	2505
	(5.19.3)	-0.6032D-13	-0.1816D-11	2847

TABLE 5.19.1

5.20 An Unstable Problem

To test our routine with an unstable problem we integrated the system

$$\underline{y}^{(1)} = A\,\underline{y} + e^{-t}\,\underline{\delta}_3 \quad , \tag{5.20.1}$$

where

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} .$$

We used the initial condition

$$\underline{y}(0) = \text{col}(2.0,\ -0.25,\ 0.5,\ -2.75) \quad . \tag{5.20.2}$$

The differential system (5.20.1) is equivalent to the equation

$$y^{(5)} = y + e^{-t}$$

which has the general solution

$$y(t) = c_1 te^{-t} + c_2 e^{t} + c_3 e^{-t} + c_4 \sin(t) + c_5 \cos(t), \tag{5.20.3}$$

where the c_i's, i=1,2,3,4,5, are constants which are determined by the initial condition. In our case (5.20.3) reduces to

$$y(t) = (t-7)e^{-t}/4 - \cos t - \sin t$$

which is free of terms containing e^t. However, small errors generated by the numerical method introduce such terms which eventually dominate the required solution.

We integrated the system (5.20.1) over the interval $0 \leq t \leq 40$ using the error tolerance ECT = 2^{-24}. The results, listed in Table 5.20.1, show the error in the first component Y_0, the Euclidean norm of the errors over all four components, and the number of steps.

Results of an Unstable Problem

t	ERROR IN Y_0	EUCLIDEAN NORM OF THE ERRORS	STEPS
10	-0.7880E-06	0.1577E-05	314
20	-0.1736E-01	0.3473E-01	634
30	0.3825E 03	0.7650E 03	1010
40	0.8425E 07	0.1685E 08	4663

TABLE 5.20.1

CHAPTER VI

CONCLUDING REMARKS

This thesis consists of a study of Nordsieck's methods for solving numerically initial value problems in differential equations. The method is developed in Chapter II. The stepsize changing is examined in Chapter III. A brief outline of an implementation of Nordsieck's methods is given in Chapter IV and the test results are reported in Chapter V. These results support the conclusion that Nordsieck's method definitely deserves consideration as a general purpose integration method. However, there are a number of items that require further investigation and research.

In particular, the area of stability in the presence of stepchanges has not been dealt with adequately. Practice has shown that large errors can result if order q formulae are used and the stepsize is not held constant for at least q steps after a change.

Because Nordsieck's method is based on polynomial approximations it appears that it is related to the problem of constructing stable polynomial filters. Hence, ideas from sequential smoothing and filter theory may lead to improvements.

Although Nordsieck's method handles first order systems of linear equations it is easy to modify the method to handle higher order equations directly. Superficially, there is some indication that solving high order equations directly may be faster than solving the equivalent linear system.

There is also the question of how to handle singularities. Clearly, if there is a singularity in the solution then polynomial methods are of little use, except for certain types of mild singularities such as a common point or an envelope for a family of solutions. Although, our experience has shown that Nordsieck's method integrates across such mild singularities it results in serious degredation of efficiency. Hence, examining special techniques to detect and deal with singularities may be rewarding. However, if such techniques are included in a general purpose program then any advantage that may be gained is partially offset by additional overhead costs which are already substantial.

As mentioned before our implementation of Nordsieck's method expends considerable effort in starting the solution. This problem is partly related to the stepchanging errors and to the accuracy of error estimates. Hence, any technique which decreases the step-changing errors produces more accurate error estimates and should improve the efficiency.

REFERENCES AND BIBLIOGRAPHY

[1] Collatz,L. (1960) The Numerical Treatment of Differential Equations, 3rd ed. Springer, Berlin.

[2] Collatz,L. (1966) Functional Analysis and Numerical Mathematics, Academic Press, New York.

[3] Descloux, J. (1963) Note on a Paper by A. Nordsieck, Dept. of Comp. Sc., Report No. 131, University of Illinois, Urbana, Ill.

[4] Gabel, G. (1968) Predictor Corrector Methods using Divided Differences, Master's Thesis, University of Toronto.

[5] Gear, C.W. (1966) The numerical Integration of Ordinary Differential Equations of Various Orders, Argonne National Lab. Report, ANL 7126.

[6] Gear, C.W. (1967) The Numerical Integration of Ordinary Differential Equations, Math. Comp.,21, pp. 146-156.

[7] Gear, C.W. (1971) Initial Value Problems in Ordinary Differential Equations, Prentice-Hall, Englewood-Cliffs, New York.

[8] Fehlberg, E. (1968) Klassische Runge-Kutta-Formeln funfter und siebenter Ordnung mit Schrittweiten-Kontrolle, Computing 4, 1969, pp. 93-106.

[9] Henrici, P. (1962) Discrete Variable Methods for Ordinary Differential Equations, John Wiley and Sons, New York.

[10] Hull, T.E. and Cremer, A.L. (1963) Efficiency of Predictor Corrector Schemes, JACM, 10, pp.291-301.

[11] Hull, T.E. et.al. (1971) Comparing Numerical Methods for Ordinary Differential Equations, Dept. of Comp. Sci., Technical Report No. 29, University of Toronto.

[12] Isaacson, E. and Keller, H.B. (1966) Analysis of Numerical Methods. John Wiley and Sons, New York.

[13] Kohfeld, J.J. and Thompson, G.T. (1968) A Modification of Nordsieck's Method Using an Off-Step Point, JACM, 15, pp. 390-401.

[14] Krough, F.T.
Report, Jet Prop. Lab., Pasadena, Calif.

[15] Lewis, H.R. and Stovall, E.J. (1965) A FORTRAN Version of Nordsieck's Scheme for the Numerical Integration of Differential Equations, Los Almos Scientific Laboratory, Report No. LA-3292,

[16] Nordsieck, A. (1962) On the Numerical Integration of Ordinary Differential Equations, Math. Comp. 16, pp. 22-49.

[17] Osborne, M.R. (1966) On Nordsieck's Method for the Numerical Solution of Ordinary Differential Equations, Bit. 6, pp. 51-57.

[18] Ralston, A. (1965) A First Course in Numerical Analysis, McGraw-Hill, New York.

[19] Rutishauser, H. (1952) Uber die Instabilitat von Methoden zur Integration gewoehnlicher Differentialgleichungen, ZAMP III 15, pp. 65-74.

APPENDIX

Program Listings

The appendix contains the program listing of the Differential Equation Solver NDSK. It is followed by an example of a calling program for NDSK.

```
$SIGN SEKE PRINT=PN FORM=BK
ON AT  14:31.45 ON TUE MAY 29/73  LAST ON AT  12:08.49
$SET LINECNT=50
$COPY *SOURCE*
C       DIFFERENTIAL EQUATION SOLVER NSDK
C
C    THE FOLLOWING PROGRAM IS A COLLECTION OF SUBROUTINES
C    WHICH SOLVE THE INITIAL VALUE PROBLEM IN THE FORM
C
C    DY(T) / DT = F(T;Y1,Y2,....,YN)  (1)
C    GIVEN YI(T0)=Y0, I=1,1,....,N.
C
      SUBROUTINE NDSK(TO,TF,TPO,HMAX,HMIN,YO,NQQ,ECQ,
     *IODR,KDEQ,CQ)
C
C    IN GENERAL, THIS SUBROUTINE ORGANIZES AND SUPERVISES THE
C    INTEGRATION PROCESS. FIRST, THE INITIAL VALUES ARE TRANS-
C    FERED INTO A COMMON BLOCK. NEXT, SOME PARAMETERS ARE
C    CHECKED FOR FAULTS. THEN, THE OTHER SUBROUTINES ARE
C    CALLED IN THE REQUIRED ORDER TO CARRY OUT THE INTEGRATION
C    PROCESS. FINALLY, THE TERMINAL VALUES OF THE SOLUTION
C    ARE PLACED IN THE ARRAY YO.
C
      DOUBLE PRECISION A(10,18),YO(10),TO,TF,TPO,HMAX,HMIN,
     *T,TQ,H,HW,TE
      DIMENSION CQ(3)
      COMMON /BLK1/A
      COMMON /BLK2/EC,ECT,IOLD,MAXR,NEQ,KDEL,EX1,EX2,EX3
C    THE PARAMETERS HAVE THE FOLLOWING SIGNIFICANCE:
C    TO       IS THE INITIAL VALUE OF THE INDEPENDENT VARIABLE.
C    TF       IS THE FINAL VALUE OF THE INDEPENDENT VARIABLE.
C    HMAX     IS THE MAXIMUM VALUE TO WHICH THE STEPSIZE MAY
C             BE INCREASED. ON DEFAULT, HMAX IS SET TO 1.0.
C    HMIN     IS THE MINIMUM VALUE TO WHICH THE STEPSIZE MAY
C             BE DECREASED. ON DEFAULT, HMIN IS SET TO 0.1D-15.
C    TPO      IS THE LENGTH OF THE INTERVAL BETWEEN
C             PRINT-POINTS.
C    YO       IS A VECTOR CONTAINING THE INITIAL VALUES OF THE
C             DEPENDENT VARIABLE. WHEN CONTROL RETURNS TO THE
C             CALLING PROGRAM YO CONTAINS THE FINAL VALUE OF
C             THE INDEPENDENT VARIABLE.
C    NQQ      IS THE NUMBER OF EQUATIONS IN THE SYSTEM ( 1 ).
C    ECQ      IS THE ERROR TOLERANCE SUPPLIED BY THE CALLING
C             PROGRAM.
C    IODR     IS THE MAXIMUM ORDER TO BE USED. THIS VALUE MUST
C             BE GREATER THAN 1 AND LESS THAN OR EQUAL TO 12.
C    KDEQ     IS THE MINIMUM NUMBER OF STEPS THAT THE PROGRAM
C             TAKES BETWEEN ORDER CHANGES.
C    CQ       IS A VECTOR OF WEIGTHS USED FOR SCALING THE
C             STEPSIZE.
```

```
C    A(10,18)  IS THE WORKING ARRAY.
C    A(N,I),I=1,2,3,....,12,IS USED TO HOLD Y(T) AND ITS FIRST
C            ELEVEN DERIVATIVES.
C    A(N,13) CONTAINS THE CORRECTION FOR A SINGLE ITERATION.
C    A(N,14) CONTAINS THE ACCUMULATED VALUE OF THE CORRECTION.
C    A(N,15) CONTAINS THE CORRECTION APPLIED AT THE
C            PREVIOUS STEP.
C    A(N,16) CONTAINS THE BACKWARD DIFFERENCE OF THE
C            CORRECTION.
C    A(N,17) CONTAINS VARIOUS PARAMETERS.
C    A(N,18) CONTAINS THE INITIAL VALUES OF Y(T).
C
C    CHECK PARAMETER LIST FOR FAULTS. IF SUCH ARE FOUND,
C    ATTEMPT TO CORRECT THESE, OR PRINT A MESSAGE AND RETURN
C    CONTROL TO THE CALLING PROGRAM. ALSO, PLACE THE PARA-
C    METERS IN THE WORKING ARRAY A.
C
    1 ECT=ECQ
      EC=ECT
      MAXR=12
      NEQ=NQQ
      KDEL=KDEQ
      DO 40 I=1,3
      IF(CQ(I).LT.(0.3).OR.CQ(I).GT.(3.0)) CQ(I)=1.0
   40 A(7+I,17)=DBLE(CQ(I))
      DO 42 N=1,NEQ
   42 A(N,18)=YO(N)
      IF(HMIN.LE.0.0) HMIN=1.0E-15
      IF(HMAX.LT.HMIN) HMAX=1.0
      IF(HMAX.GT.TPO) HMAX=TPO
      A(1,17)=TO
      A(2,17)=TF
      A(3,17)=TPO
      A(4,17)=HMAX
      A(5,17)=HMIN
      IF(NEQ.GE.1.AND.NEQ.LE.10) GO TO 50
      CALL RPRNT(0.0D 00)
      RETURN
C
   50 IF(IODR.GE.1.AND.IODR.LE.12) MAXR=IODR
C
C    CALL SUBROUTINE START TO INITIALIZE THE WORKING ARRAY FOR
C    EULER'S METHOD, AND CALL SUBROUTINE PRNT TO PRINT THE
C    INITIAL VALUES.
C
      T=TO
      CALL START(IOR,H,KSTEP,T)
      TQ=TO+DSIGN(TPO,H)
   60 CALL PRNT(0,0,IOR,H,0.0D 00,T)
C
```

```
C    INTEGRATE ONE STEP FORWARD WITH STEPSIZE H, CALL
C    SUBROUTINE TEST TO DETERMINE THE PROPER STEPSIZE HW.
C
 100 CALL NORD(1,H,T,KSTEP,IOR)
 103 CALL TEST(KSTEP,T,H,HW,IOR)
     A(6,17)=H
C
C    IF HW, AS CALCULATED BY TEST, IS GREATER THAN OR EQUAL
C    TO H THE STEP IS ACCEPTED; OTHERWISE, THE STEP IS
C    REPEATED WITH STEPSIZE HW.
C
 106 IF((DABS(H)).LE.(DABS(HW))) GO TO 120
     H=HW
C
C    IF HW IS LESS THAN HMIN THE INTEGRATION IS TERMINATED.
C
     IF((DABS(HW)).GE.HMIN) GO TO 100
 108 TO=T+A(6,17)
     CALL SPRNT(KSTEP,TO)
     GO TO 159
 120 KSTEP=KSTEP+1
     T=T+H
     H=HW
C
C    IF THE NEXT STEP PASSES THE ENDPOINT BRANCH TO
C    STATEMENT 150.
C
     TE=DABS(T-TF)
     IF(TE.LE.(DABS(H))) GO TO 150
C
C    IF THE NEXT STEP PASSES A PRINT-POINT CALL PRNT TO
C    INTERPOLATE AND TO PRINT THE SOLUTION AT THAT POINT.
C
 125 TE=DABS(T-TQ)
     IF(TE-DABS(H)) 130,135,100
 130 CALL APRNT(KSTEP,H,TE,T)
     GO TO 140
 135 CALL NPRNT(KSTEP,T)
 140 TQ=TQ+DSIGN(TPO,H)
     GO TO 100
C
C    THE FINAL VALUE OF Y(T) IS CALCULATED, PRINTED, AND
C    PLACED IN YO.
C
 150 IF(TF.LT.TO) TE=-TE
     CALL NORD(-1,TE,T,KSTEP,IOR)
     TO=T+TE
     CALL NPRNT(KSTEP,TO)
 159 DO 160 I=1,NEQ
 160 YO(I)=A(I,1)
```

```
  999 RETURN
      END
C
C
C
C
C
C
      SUBROUTINE NORD(ICALL,HN,T,KSTEP,NEWO)
C
C     THE SUBROUTINE NORD INTEGRATES THE SYSTEM          ( 1 )
C     OF ORDINARY DIFFERENTIAL EQUATIONS EXACTLY ONE STEP
C     FORWARD AT EACH CALL USING NORDSIECK'S METHOD WITH THE
C     COEFFICIENTS DEFINED IN SUBROUTINE COEFTS.
C     THE STEPSIZE HN MAY BE DIFFERENT FOR EACH CALL. THE
C     ORDER NEWO CAN VARY FROM 2 THROUGH 12, BUT SHOULD NOT
C     CHANGE BY MORE THAN 1 FOR SUCCESSIVE CALLS.
C     THE SUBROUTINE REQUIRES A DRIVING FUNCTION
C
C           FEVAL(NEQ,T,Y,F,TO,TF,HN,EC)
C
C     TO BE SUPPLIED BY THE USER TO EVALUATE THE FUNCTION
C     F OF ( 1 ).
C
      DOUBLE PRECISION A(10,18),C(3,12),ASAVE(10,16),D(10),
     *AE(10,17),Y(10),F(10),HLD,HSAVE,TH,T,HN,RHO,RH,TO,TF
      DIMENSION CP(12)
      EQUIVALENCE (A(6,17),HLD),(A(7,17),HSAVE),(A(1,17),TO),
     *(A(2,17),TF),(A(1,1),Y(1))
      COMMON /BLK1/A
      COMMON /BLK2/EC,ECT,IOLD,MAXR,NEQ,KDEL,EX1,EX2,EX3
      COMMON /BLK3/ASAVE,D
      COMMON /BLK6/C
C
C     THE PARAMETERS HAVE THE FOLLOWING SIGNIFICANCE:
C     ICALL      IS A FLAG. IF ICALL IS A POSITIVE INTEGER
C                Y(T+H) IS CALCULATED. IF ICALL EQUALS ZERO
C                THE SOLUTION IS INTERPOLATED TO A PRINT-
C                POINT. IF ICALL IS NEGATIVE THE SOLUTION IS
C                EXTRAPOLATED TO THE THE END-POINT.
C     HN         IS THE CURRENT STEPSIZE.
C     T          IS THE CURRENT VALUE OF THE INDEPENDENT VARIABLE.
C     NEWO       IS THE ORDER OF THE METHOD USED FOR THE
C                PRESENT STEP.
C     IOLD       IS THE ORDER USED FOR THE PREVIOUS STEP.
C     KSTEP      IS THE STEP COUNT.
C     KODR       IS THE VALUE OF KSTEP AT THE LAST ORDER CHANGE.
C     ASAVE      IS USED TO STORE A TO BE RETRIEVED IF THE STEP
C                IS TO BE REPEATED.
C     AE         IS USED TO STORE A WHEN THE SOLUTION IS TO BE
```

```
C                INTERPOLATED TO A PRINT-POINT.
C
      TH=T+HN
      NE=NEWO
   99 IF(ICALL) 103,100,102
C
C     A IS SAVED TO BE REINSTATED WHEN THE INTERPOLATION TO A
C     PRINT-POINT IS COMPLETED.
C
  100 DO 101 N=1,NEQ
      DO 101 I=1,17
  101 AE(N,I)=A(N,I)
      GO TO 131
C
C     CHECK WHETHER THE STEPSIZE OR THE ORDER IS TO BE CHANGED.
C
  102 IF(NEWO.NE.IOLD) CALL COEFTS(NEWO,0,C,D,CP)
      IF(DABS(HLD)-DABS(HN)) 103,103,120
C
C     A AND H ARE SAVED TO BE REINSTATED IF THE TESTS FAIL.
C
  103 HSAVE=HLD
  104 DO 110 N=1,NEQ
      ASAVE(N,15)=A(N,14)
      ASAVE(N,16)=A(N,15)
      DO 110 I=1,NEWO
  110 ASAVE(N,I)=A(N,I)
      GO TO 126
C
C     THE STEPSIZE IS TO BE REDUCED; HENCE, THE VALUES OBTAINED
C     AT THE PREVIOUS STEP ARE REINSTATED.
C
  120 HLD=HSAVE
      DO 121 N=1,NEQ
      A(N,14)=ASAVE(N,15)
      A(N,15)=ASAVE(N,16)
      DO 121 I=1,NEWO
  121 A(N,I)=ASAVE(N,I)
C
C     A CORRECTION IS CALCULATED IF H IS TO BE DECREASED.
C
      IF(NEWO.LE.6) GO TO 126
      NW=NEWO+10
      CALL COEFTS(NW,1,C,D,CP)
      DO 124 N=1,NEQ
      DO 124 J=3,NEWO
  124 A(N,J)=A(N,J)+D(J-2)*ASAVE(N,13)
C
C     THE BACKWARD DIFFERENCE OF THE HIGHEST ORDER SCALED
C     DERIVATIVE IS APPENDED TO CONTINUE THE INTEGRATION
```

```
C    WITH HIGHER ORDER.
C
 126 IF(NEWO.LE.IOLD) GO TO 130
     DO 128 N=1,NEQ
 128 A(N,NEWO)=A(N,14)/C(2,NEWO-1)
 130 IF(HN.EQ.HLD) GO TO 216
C
C    THE VALUES IN THE WORKING ARRAY A ARE SCALED FOR A
C    STEPSIZE CHANGE.
C
 131 RH=0.1D 01
     RHO=HN/HLD
     DO 132 I=2,NE
     RH=RH*RHO
     DO 132 N=1,NEQ
 132 A(N,I)=RH*A(N,I)
C
C    THE PREDICTION IS DONE BY EFFECTIVELY MULTIPLYING THE
C    VECTOR OF SCALED DERIVATIVES BY A PASCAL MATRIX.
C
 216 IF(NE.LE.1) GO TO 270
 218 DO 220 N=1,NEQ
     DO 220 I=2,NE
     DO 220 J=I,NE
     K=NE+I-J-1
 220 A(N,K)=A(N,K)+A(N,K+1)
C
C    THE SOLUTION IS CORRECTED. CORRECTIONS ARE CALCULATED
C    UNTILL THE RESIDUAL IS LESS THAN THE ERROR TOLERANCE.
C    AT MOST THREE CORRECTIONS ARE DONE PER STEP.
C
 270 DO 275 N=1,NEQ
     A(N,15)=A(N,14)
     A(N,14)=0.0D 00
 275 A(N,13)=0.0D 00
     DO 285 L=1,3
     RH=HN
 276 CALL FEVAL(TH,Y,F,TO,TF,HN,EC)
     IF(RH.NE.HN) GO TO 120
     DO 280 N=1,NEQ
     A(N,13)=HN*F(N)-A(N,2)
     Y(N)=Y(N)+C(1,1)*A(N,13)
     A(N,2)=A(N,2)+A(N,13)
 280 A(N,14)=A(N,14)+A(N,13)
     NQ=NEQ
     DO 282 N=1,NEQ
     IF(DABS(A(N,13)).LT.DBLE(EC/FLOAT(NEQ))) NQ=NQ-1
 282 CONTINUE
     IF(NQ.LE.0) GO TO 286
 285 CONTINUE
```

```
 286 IF(ICALL.EQ.0) GO TO 296
C
C    THE SCALED DERIVATIVES ARE CORRECTED.
C
      IF(NEWO.LT.3) GO TO 288
      DO 287 I=3,NEWO
      DO 287 N=1,NEQ
 287 A(N,I)=A(N,I)+C(1,I)*A(N,14)
C
C    THE BACKWARD DIFFERENCE OF THE CORRECTION TERM IS
C    CALCULATED TO BE USED TO ESTIMATE THE STEPSIZE FOR
C    THE NEXT HIGHER ORDER.
C
 288 DO 290 N=1,NEQ
      ASAVE(N,13)=A(N,14)
      A(N,14)=C(1,NEWO)*A(N,14)
 290 A(N,16)=A(N,14)-A(N,15)
 294 IOLD=NEWO
      RETURN
C
C    THE SOLUTION AT THE PRINT-POINT IS PUT INTO A(N,18).
C
 296 DO 298 N=1,NEQ
      A(N,18)=A(N,1)
      DO 298 I=1,17
 298 A(N,I)=AE(N,I)
 999 RETURN
      END
C
C
C
C
C
C
      SUBROUTINE TEST(KSTEP,T,HLD,HW,IOR)
C
C    SUBROUTINE TEST CHECKS THE STEPSIZE AND THE ORDER.
C
      DOUBLE PRECISION A(10,18),C(3,12),WTS(10),T,HLD,HW,
     *RHO,T1,T2,T3
      DIMENSION CP(12)
      COMMON /BLK1/A
      COMMON /BLK2/EC,ECT,IOLD,MAXR,NEQ,KDEL,EX1,EX2,EX3
      COMMON /BLK4/WTS,CP,KODR,KHC
      COMMON /BLK6/C
C
C    THE PARAMETERS HAVE THE FOLLOWING SIGNIFICANCE:
C    HW          IS THE NEW STEPSIZE.
C    HLD         IS THE STEPSIZE WHICH HAS BEEN USED.
C
```

```
C    CHECK WHETHER THE ERROR CONSTANT HAS BEEN CHANGED BY
C    FEVAL. IF SO, THE NEW ERROR CONSTANT IS PLACED IN CP.
C    THE MAXIMUM VALUE OF THE SOLUTION IS PLACED IN WTS.
C
  60 HW=HLD
     DO 70 N=1,NEQ
  70 WTS(N)=DMAX1(WTS(N),DABS(A(N,1)))
     IF(ECT.EQ.EC) GO TO 130
     DO 80 I=1,12
  80 CP(I)=CP(I)*(ECT/EC)**2
     ECT=EC
C
C    A CHECK IS DONE TO DETERMINE WHETHER THE STEPSIZE HAS
C    BEEN REDUCED MORE THAN FOUR TIMES ON THE PRESENT STEP.
C    KQ CONTAINS THE COUNT. IF SO, THE ERRORS IN THE STORED
C    QUANTITIES ARE ASSUMED TO HAVE EXCEEDED THE ACCEPTABLE
C    LEVEL AND THE INTEGRATION IS RESTARTED WITH EULER'S
C    METHOD.
C
 130 IF(KSTEP.NE.KHC.OR.KSTEP.LE.2) GO TO 140
     KQ=KQ+1
     IF(KQ.LE.4.OR.IOR.LE.2) GO TO 200
 137 IOR=2
     CALL OPRNT(KSTEP,IOR,T)
     CALL RSTART(HW)
     IF(DABS(HW).GE.DABS(HLD)) HW=0.95*HLD
     RETURN
 140 KQ=0
C
C    THE QUANTITIES REQUIRED TO CALCULATE HW ARE ACCUMULATED.
C    T2 IS A MULTIPLIER SUCH THAT H/T2 IS THE REQUIRED HW.
C
 200 RHO=0.0
     DO 205 N=1,NEQ
 205 RHO=RHO+(A(N,14)/WTS(N))**2
 206 T2=A(9,17)*(CP(IOR)*RHO*C(3,IOR-1)**2)**EX2
C
C    THE ORDER IS CHECKED ONLY EVERY KDEL STEPS.
C
     IF(KSTEP-KODR-KDEL) 210,210,220
 210 IF(T2.GT.(0.8D 00).AND.T2.LT.(0.10D 01)) RETURN
     GO TO 300
C
C    THE QUANTITIES T1 AND T3, REQUIRED TO FIND HW FOR THE
C    NEXT LOWER OR HIGHER ORDER, ARE ACCUMULATED.
C
 220 T1=0.0D 00
     T3=0.0D 00
     KODR=KSTEP
     DO 240 N=1,NEQ
```

```
      T1=T1+(A(N,IOR)/WTS(N))**2
  240 T3=T3+(A(N,16)/WTS(N))**2
      RHO=0.1D 15
      IF(IOR.GT.2) RHO=A(8,17)*(CP(IOR-1)*T1*C(3,IOR-1)**2)
     ***EX1
      T1=RHO
      RHO=0.1D 15
      IF(IOR.LT.MAXR) RHO=A(10,17)*(CP(IOR+1)*T3*C(3,IOR-1)
     ***2)**EX3
      T3=RHO
      RHO=DMIN1(T1,T2,T3)
C
C     RHO IS SUCH THAT H/RHO IS THE REQUIRED STEPSIZE.
C     IF RHO EQUALS T1 THE ORDER IS LOWERED. IF RHO EQUALS T3
C     THE ORDER IS INCREASED.
C
      IF(RHO.GE.(0.1D 01)) GO TO 300
      IF(DABS(T2-RHO)-0.10*T2) 210,210,250
  250 T2=RHO
      IF(T2.EQ.T1) IOR=IOR-1
      IF(T2.EQ.T3) IOR=IOR+1
      IOR=MAX0(IOR,1)
      EX1=1.0/(2.0*FLOAT(IOR))
      EX2=1.0/(2.0*(1.0+FLOAT(IOR)))
      EX3=1.0/(2.0*(2.0+FLOAT(IOR)))
C
C     THE NEW STEPSIZE HW IS CALCULATED.
C
  300 HW=0.9D 00*HLD/DMAX1(T2,0.1D 00)
      HW=DSIGN(DMIN1(A(4,17),DABS(HW)),HW)
  305 IF(DABS(HW).LT.DABS(HLD)) KHC=KSTEP
  345 RETURN
      END
C
C
C
C
C
C
      SUBROUTINE START(IOR,H,KSTEP,T)
C
C     SUBROUTINE START INITIALIZES THE WORKING ARRAY.
C     THE MEANING OF THE ARGUMENTS IS THE SAME AS BEFORE.
C
      DOUBLE PRECISION A(10,18),C(3,12),ASAVE(10,16),D(10),
     *Y(10),F(10),WTS(10),H,TO,TF,TPO,HMAX,HMIN,HLD,T
      DIMENSION CP(12)
      EQUIVALENCE (A(1,1),Y(1))
      EQUIVALENCE (A(1,17),TO),(A(2,17),TF),(A(4,17),HMAX),
     *(A(6,17),HLD)
```

```
      COMMON /BLK1/A
      COMMON /BLK2/EC,ECT,IOLD,MAXR,NEQ,KDEL,EX1,EX2,EX3
      COMMON /BLK3/ASAVE,D
      COMMON /BLK4/WTS,CP,KODR,KHC
      COMMON /BLK6/C
C
      IOR=2
      KSTEP=0
      DO 102 I=1,10
      D(I)=0.0D 00
  102 WTS(I)=0.1D 01
C
C     THE INITIAL VALUES, AS PROVIDED BY THE CALLING PROGRAM,
C     ARE PLACED IN THE WORKING ARRAY.
C
      DO 105 N=1,NEQ
      A(N,1)=A(N,18)
  105 ASAVE(N,1)=A(N,1)
C
C     THE ERROR CONSTANT TO BE USED INITIALLY IS CALCULATED.
C
      CALL COEFTS(IOR,-1,C,D,CP)
      C(2,1)=0.1D 01
      C(3,1)=0.1D 01
      DO 110 I=2,12
      C(1,I)=0.0D 00
      C(2,I)=0.1D 01+C(2,I-1)
      CP(I)=(CP(I)/ECT)**2
  110 C(3,I)=C(3,I-1)*C(2,I)
      C(1,2)=0.1D 01
C
C     THE REQUIRED PREPARATIONS ARE MADE HERE TO START THE
C     INTEGRATION PROCESS WITH THE CLASSICAL EULER METHOD
C     USING THE PRESENT VALUES OF THE VARIABLES, DEPENDENT
C     AND INDEPENDENT, AS THE INITIAL VALUES.
C
      ENTRY RSTART(H)
C
      IOLD=2
      KODR=KSTEP
      KHC=KSTEP
      H=A(5,17)**0.25
      IF(TF.LT.TO) H=-H
      HLD=H
      A(7,17)=H
      CALL FEVAL(T,Y,F,TO,TF,H,EC)
      DO 275 N=1,NEQ
      A(N,2)=H*F(N)
      DO 275 I=3,12
      A(N,I)=0.0D 00
```

```
 275 ASAVE(N,I)=0.0D 00
     EX1=0.25
     EX2=1.0/6.0
     EX3=0.125
     CALL COEFTS(IOR,0,C,D,CP)
     RETURN
     END
C
C
C
C
C
C
     SUBROUTINE COEFTS(IODR,IX,C,D,CP)
C    SUBROUTINE COEFTS ASSIGNS THE PROPER COEFFICIENTS TO BE
C    USED FOR THE IODR'TH ORDER METHOD.
     DOUBLE PRECISION C(3,12),D(10)
     DIMENSION CP(12)
C
     IF(IX) 300,100,200
C
 100 GO TO (1,2,3,4,5,6,7,8,9,10,11,12),IODR
   1 C(1,1)=0.1D 01
     RETURN
   2 C(1,1)=0.5D 00
     RETURN
   3 C(1,1)=0.5D 00
     C(1,3)=0.5D 00
     RETURN
   4 C(1,1)=0.4166666666666667D 00
     C(1,3)=0.75D 00
     C(1,4)=0.1666666666666667D 00
     RETURN
   5 C(1,1)=0.375D 00
     C(1,3)=0.9166666666666667D 00
     C(1,4)=0.33333333333333D 00
     C(1,5)=0.4166666666666667D-01
     RETURN
   6 C(1,1)=0.34861111111111111D 00
     C(1,3)=0.1041666666666667D 01
     C(1,4)=0.48611111111111111D 00
     C(1,5)=0.1041666666666667D 00
     C(1,6)=0.8333333333333333D-02
     RETURN
   7 C(1,1)=0.32986111111111111D 00
     C(1,3)=0.1141666666666667D 01
     C(1,4)=0.625D 00
     C(1,5)=0.17708333333333333D 00
     C(1,6)=0.25D-01
     C(1,7)=0.13888888888888889D-02
```

```
      RETURN
    8 C(1,1)=0.31559193121693l2D 00
      C(1,3)=0.1225D 01
      C(1,4)=0.7518518518518518D 00
      C(1,5)=0.25520833333333330 00
      C(1,6)=0.4861111111111111D-01
      C(1,7)=0.4861111111111111D-02
      C(1,8)=0.1984126984126984D-03
      RETURN
    9 C(1,1)=0.3042245370370370D 00
      C(1,3)=0.1296428571428571D 01
      C(1,4)=0.8685185185185185D 00
      C(1,5)=0.3357638888888889D 00
      C(1,6)=0.7777777777777778D-01
      C(1,7)=0.1064814814814815D-01
      C(1,8)=0.7936507936507936D-03
      C(1,9)=0.2480158730158730D-04
      RETURN
   10 C(1,1)=0.2948680004409171D 00
      C(1,3)=0.1358928571428571D 01
      C(1,4)=0.9765542328042328D 00
      C(1,5)=0.4171875D 00
      C(1,6)=0.1113541666666667D 00
      C(1,7)=0.1875D-01
      C(1,8)=0.1934523809523809D-02
      C(1,9)=0.1116071428571428D-03
      C(1,10)=0.2755731922398580-05
      RETURN
   11 C(1,1)=0.2869754464285714D 00
      C(1,3)=0.1414484126984126D 01
      C(1,4)=0.1077215608465608D 01
      C(1,5)=0.4985670194003505D 00
      C(1,6)=0.1484375D 00
      C(1,7)=0.2906057098765427D-01
      C(1,8)=0.3720238095238095D-02
      C(1,9)=0.2996858465608465D-03
      C(1,10)=0.1377865961199294D-04
      C(1,11)=0.2755731922398589D-06
      RETURN
   12 C(1,1)=0.2801895964439367D 00
      C(1,3)=0.1464484126984123D 01
      C(1,4)=0.1171514550264546D 01
      C(1,5)=0.5793581900352713D 00
      C(1,6)=0.1883228615520[illegible]5D 00
      C(1,7)=0.4143036265432[illegible]D 01
      C(1,8)=0.6211144179894[illegible]D-02
      C(1,9)=0.6252066798941796D-03
      C(1,10)=0.4041740152851263D-04
      C(1,11)=0.1515652557319224D-05
      C(1,12)=0.2505210838544175D-07
```

```
      RETURN
C
  200 GO TO (16,17,18,19,20,21,22),IODR
   16 D(1)=0.68247126436781650-01
      D(2)=0.77825670498084280-01
      D(3)=0.49928160919540230-01
      D(4)=0.17097701149425290-01
      RETURN
   17 D(1)=0.48415977961432510-01
      D(2)=0.67148760330578510-01
      D(3)=0.56387741046831950-01
      D(4)=0.28719008264462810-01
      D(5)=0.82529843893480250-02
      RETURN
   18 D(1)=0.32694892473118280-01
      D(2)=0.53419952210274790-01
      D(3)=0.55422547043010750-01
      D(4)=0.37438396057347670-01
      D(5)=0.16109431003584230-01
      D(6)=0.40314580133118520-02
      RETURN
   19 D(1)=0.21279466060778190-01
      D(2)=0.40022826069214300-01
      D(3)=0.49431880966026150-01
      D(4)=0.41594875193284740-01
      D(5)=0.23835689566305870-01
      D(6)=0.89447442330146150-02
      D(7)=0.19880222632459210-02
      RETURN
   20 D(1)=0.13452733164271630-01
      D(2)=0.28622727581060910-01
      D(3)=0.41008690828402370-01
      D(4)=0.41310301610782380-01
      D(5)=0.29567307692307690-01
      D(6)=0.14790991593876210-01
      D(7)=0.49308564032121720-02
      D(8)=0.98619057620125990-03
      RETURN
   21 D(1)=0.83057678970786550-02
      D(2)=0.19721543638455760-01
      D(3)=0.32155833569268360-01
      D(4)=0.37727816642120780-01
      D(5)=0.32386322743746850-01
      D(6)=0.20248541856605170-01
      D(7)=0.90000165181574720-02
      D(8)=0.27000423276094250-02
      D(9)=0.49091788140736760-03
      RETURN
   22 D(1)=0.50282849835984020-02
      D(2)=0.13180334341267250-01
```

```
      D(3)=0.24099079777717065D-01
      D(4)=0.32275141316955920D-01
      D(5)=0.32311109117861330D-01
      D(6)=0.24239419406420250D-01
      D(7)=0.13467035943516190D-01
      D(8)=0.53868657156215640D-02
      D(9)=0.14691475231017240D-02
      D(10)=0.24485797623601650D-03
      RETURN
C
C
C     THE COEFFICIENTS OF THE LOCAL TRUNCATION ERROR TERMS ARE
C     PLACED IN CP. CP(P) WILL CONTAIN THE COEFFICIENT FOR THE
C     P'TH ORDER METHOD.
C
  300 CP(1)=0.5
      CP(2)=0.08334
      CP(3)=0.04167
      CP(4)=0.02639
      CP(5)=0.01875
      CP(6)=0.01427
      CP(7)=0.01137
      CP(8)=0.00936
      CP(9)=0.00789
      CP(10)=0.00678
      CP(11)=0.00527
      CP(12)=0.00412
      RETURN
      END
C
C
C
      SUBROUTINE PRNT(IPX,KSTEP,IORDER,H,HP,T)
C
C     SUBROUTINE PRNT IS A COLLECTION OF ALL WRITE AND THEIR
C     ASSOCIATED FORMAT STATEMENTS. TO AVOID MULTIPLE TESTS,
C     SEPARATE ENTRY POINTS ARE PROVIDED FOR MOST WRITE STATE-
C     MENTS. THE EXPLANATIONS ARE OMITTED. THESE CAN BE
C     OBTAINED BY CHECKING THE ASSOCIATED FORMAT STATEMENTS.
C     IPX   IS A FLAG. IF IPX=0 Y,T, AND KSTEP ARE PRINTED.
C           IF IPX=1 THE SOLUTION IS INTERPOLATED TO A
C           PRINT-POINT. IF IPX=-1 THE INTEGRATION TERMINATES.
C     HP    IS THE DIFFERENCE BETWEEN THE CURRENT VALUE OF
C           T AND THE NEXT PRINT-POINT.
C
      DOUBLE PRECISION A(10,18),Y(10),F(10),YP(10),T,H,HP,TZ
      COMMON /BLK1/A
      COMMON /BLK2/EC,ECT,IOLD,MAXR,NEQ,KDEL,EX1,EX2,EX3
      EQUIVALENCE (A(1,1),Y),(A(1,2),F),(A(1,18),YP)
C
```

```
  950 FORMAT(' ',I6,3X,F12.5,3X,5(D19.12,1X)/27X,5(D19.12,1X))
  951 FORMAT(' INTEGRATION TERMINATED,H HAS BECOME TO SMALL')
  952 FORMAT('    PARAMTERS ARE IMPROPERLY SPECIFIED,INTE-
     *GRATION NOT STARTED  ')
  953 FORMAT('0THE OUTPUT COLUMNS ARE    '//'  KSTEP T Y(1)
     *Y(2)        ...     Y(NEQ) '//'0  ')
  954 FORMAT('    AT START T =  ',F12.5,10X,'STEPSIZE = ',
     *F12.5,5X,'ORDER = ',I6//)
  955 FORMAT('0',30X,'AT T = ',E12.5,5X,'THE ORDER CHANGED
     *TO  ',I6,5X,'KSTEP = ',I6//)
  956 FORMAT('0',30X,'AT T = ',E12.5,5X,'H CHANGED FROM  ',
     *E12.7,5X,'TO  ',E12.7,5X,'KSTEP = ',I6//)
  957 FORMAT('0  THE INTEGRATION IS RESTARTED USING EULER'S
     *METHOD  # OF STEPS ',I6,' ORDER ',I6,'  T  ',D20.13,'
     *  H ',D20.13//
C
      NQ=NEQ
      IF(KSTEP.GT.0) GO TO 4
      WRITE(6,953)
      WRITE(6,954) T,H,IORDER
    4 IF(IPX) 20,30,10
C
      ENTRY RPRNT(T)
    5 WRITE(6,952)
      RETURN
C
      ENTRY APRNT(KSTEP,H,HP,T)
   10 IF(H.LT.(0.0D 00)) HP=-DABS(HP)
      IORDER=IOLD
      CALL NORD(0,HP,T,KSTEP,IORDER)
      TZ=HP+T
      WRITE(6,950) KSTEP,TZ,(YP(I),I=1,NQ)
      RETURN
C
      ENTRY OPRNT(KSTEP,IORDER,T)
   12 WRITE(6,955) T,IORDER,KSTEP
      RETURN
C
      ENTRY HPRNT(KSTEP,H,HP,T)
   15 WRITE(6,956) T,H,HP,KSTEP
      RETURN
C
      ENTRY TPRNT(KSTEP,IORDER,T,H)
   16 WRITE(6,957) KSTEP,IORDER,T,H
      GO TO 30
C
      ENTRY SPRNT(KSTEP,T)
   20 WRITE(6,951)
C
      ENTRY NPRNT(KSTEP,T)
```

```
30 TZ=T
   WRITE(6,950) KSTEP,TZ,(Y(I),I=1,NQ)
99 RETURN
   END
```

```
$COPY *SOURCE*
C     SAMPLE PROGRAM TO CALL NDSK
C
C     CALLING PROGRAM TO INTEGRATE THE
C
C        SINE - COSINE DIFFERENTIAL EQUATION
C
      DOUBLE PRECISION YO(10),TO,TF,TPO,HMAX,HMIN
      DIMENSION CQ(3)
      COMMON /BLK/NOF,ECT
  905 FORMAT('0 THE # OF FUNCTION EVALUATIONS = ',I6,5X,E10.5)
      TO=0.0D 00
      TF=0.2D 03
      TPO=0.1D 02
      HMAX=0.5D 01
      HMIN=0.1D-12
      YO(1)=0.0D 00
      YO(2)=0.1D 01
      NEQ=2
      ECT=0.1E-06
      IODR=10
      KDEL=0
      CQ(1)=1.1
      CQ(2)=1.2
      CQ(3)=1.3
      NOF=0
      CALL NDSK(TO,TF,TPO,HMAX,HMIN,YO,NEQ,ECT,IODR,KDEL,CQ)
      WRITE(6,905) NOF,ECT
      STOP
      END
C
      SUBROUTINE FEVAL(T,Y,F,TO,TF,HN,EC)
      DOUBLE PRECISION F(10),Y(10),TO,TF,HN,T
      COMMON /BLK/NOF,ECT
      NOF=NOF+1
      F(1)=Y(2)
      F(2)=-Y(1)
      RETURN
      END
```